МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Кваліфікаційна робота

на тему:

Розроблення програмного забезпечення для моніторінгу інформації на платформі Discord з використанням API

Рівень вищої освіти другий (магістерський)
Спеціальності 122 Комп'ютерні науки
Освітня програма Комп'ютерні науки

Виконав: студент групи МгІТ-1-23
Лапа В.С.
Науковий керівник:
к.т.н., доц. Астістова Т.І.
Рецензент
к.т.н., доц.Мельник Г.В.

Київ 2024

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет мехатроніки та комп'ютерних технологій

Кафедра комп'ютерних наук

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри КН

_____Наталія ЧУПРИНКА

«_____»_____2024 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТА

Лапи Валентину Сергійовичу

1.Тема роботи: Розроблення програмного забезпечення для моніторінгу інформації на платформі Discord з використанням API, науковий керівник роботи Астістова Тетяна Іванівна, к.т.н., доц., затверджені наказом закладу вищої освіти від 03.09.2024 року, № 118-уч.

2. Строк подання студентом роботи 22.11. 2024 р.

3. Вихідні дані до роботи:

Розробка кафедри комп'ютерних наук

4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)

Розділ 1 Аналіз предметної області; Розділ 2. Проектування програмної реалізації; Розділ 3. Розробка програмного забезпечення

Додатки - програмні коди системи, презентація.

5 . Дата видачі завдання: 03. 09. 2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	23.08.2024	
2	Розділ 1 Аналіз предметної області	03.09.2024	
3	Розділ 2. Проектування програмної реалізації	13.10.2024	
4	Розділ 3. Розробка програмного забезпечення	28.10.2024	
5	Висновки	31.10.2024	
6	Оформлення кваліфікаційної роботи (чистовий варіант)	13.10.2024	
7	Подача кваліфікаційної роботи науковому керівнику для відгуку	15.11.2024	
8	Подача кваліфікаційної роботи для рецензування	19.11.2024	
9	Перевірка кваліфікаційної роботи на наявність ознак плагіату	20.11.2024	
10	Подання кваліфікаційної роботи на затвердження завідувачу кафедри	22.11.2024	

Студент _____ Валентин ЛАПА

Науковий керівник роботи _____ Тетяна АСТІСТОВА

АНОТАЦІЯ

Лапа В.С. Розроблення програмного забезпечення для моніторингу інформації на платформі Discord з використанням API.

Дипломна магістерська робота за спеціальністю 122 «Комп'ютерні науки». — Київський національний університет технологій та дизайну, Київ, 2024 рік.

Дипломна робота присвячена аналізу використання сучасних технологій для інформування громадян про повітряні тривоги. Під час роботи було розроблено Discord-бота, який забезпечує моніторинг інформації про повітряні тривоги в Україні і повідомляє користувачів платформи в режимі реального часу.

Для вирішення поставленої задачі було створено програмне забезпечення, що дозволяє автоматизувати оповіщення про тривоги, розширити можливості оперативного оповіщення громадян і забезпечити доступність таких оповіщень на популярній платформі Discord.

Для реалізації поставленого завдання було використано мову програмування C#. Щоб отримати найновішу інформацію я використав підключення до відкритого API повітряних тривог. Функціонал бота включає можливість налаштування для різних серверів і каналів, надсилання текстових повідомлень, а також активацію голосових оповіщень у вибраних каналах. Збереження налаштувань каналів здійснюється у форматі JSON, що забезпечує зручність обробки та переносу даних.

В рамках проєкту проаналізовано актуальність впровадження технологій оповіщення на основі API, вивчена можливість використання Discord в якості платформи для впровадження систем оповіщення та оцінена ефективність інтеграції цих рішень для підвищення безпеки громадян.

Ключові слова: discord-бот, повітряна тривога, API, C#, JSON, моніторинг в режимі реального часу, автоматизація.

ANNOTATION

Lapa V.S. Development of software for monitoring information on the Discord platform using API

Master's thesis in the speciality 122 ‘Computer Science’. — Kyiv National University of Technology and Design, Kyiv, 2024.

The thesis is devoted to the analysis of the use of modern technologies to inform citizens about air alerts. During the work, a Discord bot was developed that monitors information about air alerts in Ukraine and notifies platform users in real time.

To solve this problem, we created software that automates alerts, enhances the ability to promptly notify citizens and ensures the availability of such alerts on the popular Discord platform.

The C# programming language was used to implement the task. To get the latest information, I used a connection to the open air raid alarm API. The bot's settings and functionality include the ability to configure it for different servers and channels, send text messages, and activate voice alerts in selected channels. The channel settings are stored in JSON format, which ensures easy data processing and transfer.

The project will analyze the relevance of implementing API-based alert technologies, explore the possibility of using Discord as a platform for implementing alert systems, and evaluate the effectiveness of integrating these solutions to improve the safety of citizens.

Keywords: discord bot, air raid alert, API, C#, JSON, real-time monitoring, automation.

ЗМІСТ

ВСТУП

Актуальність теми .Сучасний світ невпинно змінюється під впливом інформаційних технологій, які знаходять застосування в усіх сферах життя. Одна з важливих функцій інформаційних систем полягає в оперативному інформуванні населення про події, що мають критичне значення. Зокрема, в умовах війни та зростання загрози для безпеки громадян України, питання швидкого та ефективного повідомлення про повітряні тривоги є надзвичайно актуальним.

Платформа Discord, яка стала популярною серед користувачів завдяки своїй функціональності та універсальності, відкриває широкі можливості для автоматизації завдань, у тому числі тих, що пов’язані з розповсюдженням важливої інформації. Discord надає API для інтеграції зовнішніх сервісів та створення ботів, здатних взаємодіяти із серверами платформи. Ця технологія дозволяє розробляти інструменти, що автоматично аналізують дані й надсилають повідомлення в режимі реального часу.

Існуючі приклади реалізації Discord-ботів демонструють, що такі інструменти можуть бути не лише корисними, але й соціально значущими. Створення бота для оперативного інформування про повітряні тривоги є важливим кроком у розширенні можливостей сучасних інформаційних систем і слугує ефективним інструментом для збереження життя та здоров'я людей.

Об'єкт дослідження. Розробка програмного забезпечення для моніторингу інформації на платформі Discord, зокрема створення чат-бота, який здійснює автоматичне сповіщення про повітряні тривоги.

Мета дослідження. Створення багатофункціонального Discord-бота, здатного інтегруватися з сервісами для отримання актуальної інформації про повітряні тривоги та забезпечувати своєчасне її розповсюдження серед користувачів платформи.

Завдання.

У рамках дипломної роботи вирішуються такі завдання:

- Інтеграція API сервісу *alerts.in.ua* для отримання актуальної інформації про повітряні тривоги.
- Розробка логіки обробки запитів та отримання даних про тривоги в режимі реального часу.
- Налаштування механізмів автоматичного надсилання сповіщень до текстових каналів Discord.
- Розробка функціоналу для подання інформації через голосові канали Discord.
- Створення зручного інтерфейсу керування ботом, який дозволяє налаштовувати параметри роботи відповідно до потреб користувачів.
- Забезпечення надійності та безпеки роботи бота, зокрема захисту даних користувачів.

Наукова новизна. Запропонований проект поєднує сучасні технології інтеграції з API та автоматизацію обробки даних у реальному часі. Новизна рішення полягає у використанні платформи Discord як ефективного засобу для розповсюдження інформації критичного значення. Важливим аспектом роботи є впровадження автоматизації моніторингу та сповіщень, що забезпечує оперативність передачі інформації та мінімізує ризик людської помилки. Запропонований проект має значний соціальний та технологічний потенціал. Створений бот є прикладом використання сучасних інструментів програмування для вирішення актуальних проблем суспільства

Апробація результатів

Робота була представлена на VIII Міжнародній науково-практичної конференції конференції MSIE-2024

1. Astistova T.I., Lapa V.S. Development of software for monitoring information on the Discord platform using API / Лапа В.С., Астістова Т.І. Моніторінг інформації на платформі Discord з використанням API // VIII Міжнародна конференція «Мехатронні системи:інновації та інжиніринг– КНУТД, 2024 – С 175

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Платформа Discord та її можливості

Discord — це додаток для голосового, відео та текстового чату, яким користуються десятки мільйонів людей віком від 13 років, щоб спілкуватися зі своїми спільнотами та друзями[1]. Логотиппредставлено на рис.1.1.

Рисунок 1.1. — Логотип Discord

Люди використовують Discord щодня, щоб поговорити про багато речей - від мистецьких проектів та сімейних подорожей до домашнього завдання та підтримки психічного здоров'я. Це домівка для спільнот будь-якого розміру, але найбільш широко використовується невеликими та активними групами людей, які регулярно спілкуються.

Переважна більшість серверів — це приватні, доступні лише за запрошеннями простори для груп друзів і спільнот, які підтримують зв'язок і проводять час разом. Існують також більші відкриті спільноти, зазвичай зосереджені навколо конкретних тем, таких як популярні ігри. Користувачі мають контроль над тим, з ким вони взаємодіють і яким є їхній досвід на Discord.

Люди люблять Discord, тому що він є домом для всіх їхніх спільнот і груп друзів. Це місце, де вони можуть бути самими собою і проводити час з іншими людьми, які поділяють їхні інтереси та хобі. Розмови на Discord визначаються

людьми, яких ви обираєте, і темами, які ви обираєте для обговорення. Discord має свій власний словниковий запас.

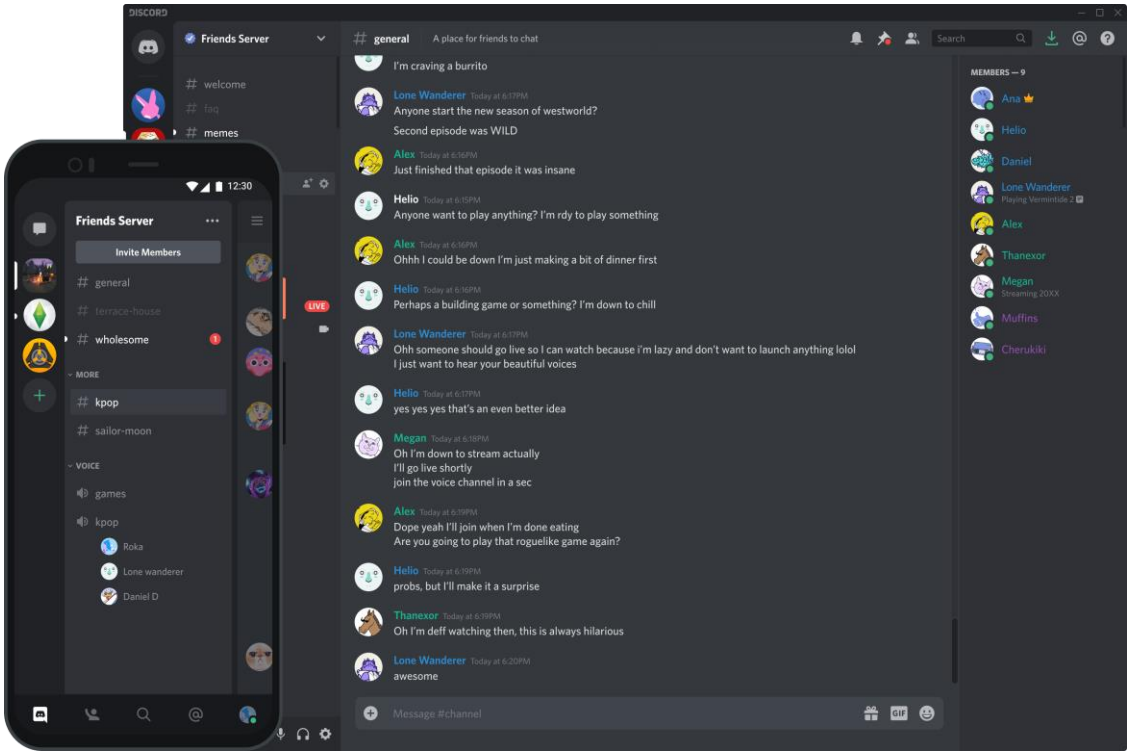


Рисунок 1.2. — Інтерфейс Discord

Розглянемо складові Discord.

Сервер: Сервери — це місця на Discord. Вони створюються певними спільнотами та групами друзів. Переважна більшість серверів невеликі і доступні лише за запрошеннями. Деякі більші сервери є загальнодоступними. Будь-який користувач може створити новий сервер безкоштовно і запросити на нього своїх друзів.

Канал: Сервери Discord організовані в текстові та голосові канали, які зазвичай присвячені певним темам і можуть мати різні правила.

- У текстових каналах користувачі можуть публікувати повідомлення, завантажувати файли та ділитися зображеннями, які можуть побачити інші в будь-який час.

- У голосових каналах користувачі можуть з'єднуватися за допомогою голосового або відеодзвінка в режимі реального часу, а також ділитися своїм екраном з друзями - ми називаємо це Go Live.

DM і GDM: Користувачі можуть надсилати приватні повідомлення іншим користувачам у вигляді прямих повідомлень (DM), а також розпочати голосовий або відеодзвінок. Більшість DM — це розмови один на один, але користувачі мають можливість запросити до розмови до дев'яти інших користувачів, щоб створити приватне групове DM (GDM), максимальний розмір якого не може перевищувати десять осіб. Групові чати не є загальнодоступними і потребують запрошення від когось із учасників групи, щоб приєднатися до них.

Прямий ефір: користувачі можуть показувати свій екран іншим людям, які перебувають з ними на сервері або в DM.

Nitro: Nitro — це преміум-сервіс підписки на Discord. Він пропонує особливі привілеї для передплатників, такі як можливість налаштувати свій тег Discord, можливість використовувати власні емоції на кожному сервері, вищий ліміт на завантаження файлів і знижку на серверні підсилювачі.

Boost серверу: Якщо ви є великим шанувальником спільноти, ви можете підвищити потужність сервера спільноти (або свого власного). Як і Nitro, Boost серверу надає серверам особливі привілеї, такі як більше власних емоцій, кращу якість відео та звуку, а також можливість встановити власне посилання-запрошення. Boost серверу можна придбати разом з Nitro або окремо.

Студентські хаби: Студентські хаби Discord дозволяють студентам підтвердити свій обліковий запис Discord за допомогою офіційної студентської електронної пошти і розблокувати доступ до ексклюзивного хабу для студентів свого навчального закладу. В межах хабу вони можуть з'єднуватися з іншими перевіреними студентами, знаходити сервери для навчальних груп або класів, а

також ділитися своїми власними серверами, до яких можуть приєднатися інші студенти. Хаби не пов'язані зі школою або її працівниками і не управляються ними. Серверами в хабі керують студенти, але можуть користуватися і ті, хто не є студентами.

1.2. Інтерфейс прикладного програмування

API, або інтерфейс прикладного програмування, — це набір правил і протоколів для створення та взаємодії з програмними додатками. API діють як посередник, дозволяючи двом різним програмам спілкуватися між собою. Цей зв'язок може включати надсилання та отримання даних, або ж дозволяє різним програмним компонентам взаємодіяти та виконувати завдання[2].

Визначаючи чіткий набір методів та інструментів, API гарантують, що різне програмне забезпечення може надійно взаємодіяти незалежно від їхньої базової архітектури чи технології. Завдяки цьому API відіграли вирішальну роль у розвитку інтернету, уможлививши створення хмарних сервісів, розробку мобільних додатків та інтеграцію веб-сервісів. Сьогодні API — це набагато більше, ніж просто бібліотеки коду. Це складні інструменти, які визначають спосіб нашої взаємодії з технологіями.

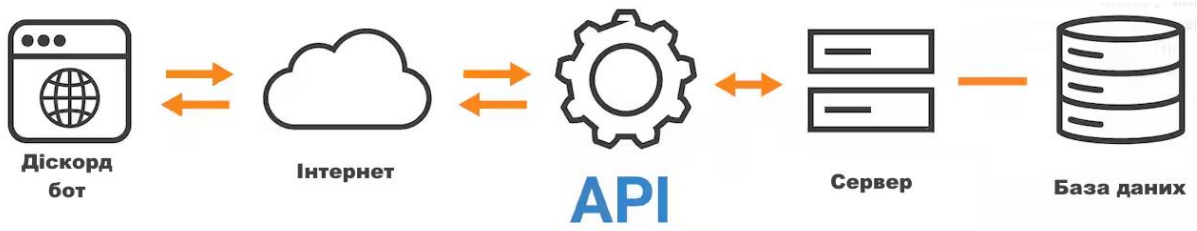


Рисунок 1.3. — Механізм обміну інформацією через API

API працюють, відкриваючи обмежену кількість дій і точок даних, з якими може взаємодіяти зовнішнє програмне забезпечення. Коли програмна система хоче отримати доступ до ресурсу, наданого іншою системою (наприклад, до даних або функціоналу), вона надсилає запит з детальним

описом дій, які їй потрібно виконати. Цей запит робиться через API. Якщо API авторизований, система обробляє цей запит і надсилає відповідь.

API часто розроблені таким чином, щоб запускатися бізнес-подіями. Подія — це будь-яка дія або зміна стану, важлива для бізнесу, наприклад, коли хтось проводить кредитною карткою, реєструється на рейс, скидає пароль або коли оновлюються запаси на складі. Таким чином, API часто використовуються в архітектурах, керованих подіями для полегшення наскрізних процесів, де доступ до декількох систем здійснюється для виконання конкретних завдань, пов'язаних з процесом.

Основні компоненти та структура API включають кінцеві точки, методи, запити та відповіді. Кінцеві точки — це конкретні адреси (URL-адреси для веб-API), за якими можна отримати доступ до API. Методи — це допустимі дії (наприклад, GET, POST, PUT, DELETE для HTTP API), які можуть бути виконані на цих кінцевих точках. Запити включають необхідні дані та параметри для виконання дії, а відповіді — це дані, які повертає API[3].

Сьогодні API в основному розробляються відповідно до схеми, яка визначає правила взаємодії API і те, як API форматується, перевіряється і документується. Структуровані для забезпечення безпечного, надійного та ефективного зв'язку між системами, API включають специфікації для процедур, структур даних, класів об'єктів та змінних.

Синхронні та асинхронні API відносяться до різних підходів у тому, як програмні системи обробляють запити та відповідають на них. Ці терміни зазвичай використовуються в контексті програмування та веб-розробки.

З синхронними API, коли надходить запит, програма блокує його і чекає на завершення операції, перш ніж перейти до наступного завдання. Це означає, що програма «синхронізується» з операцією, і вона не продовжить роботу, поки

не завершиться запитувана дія. Синхронні API часто використовуються, коли простота і читабельність коду мають вирішальне значення.

Давайте розглянемо дві найпопулярніші архітектури API, а саме REST API та SOAP API.

1. SOAP API

SOAP розшифровується як Simple Object Access Protocol — це офіційний протокол обміну повідомленнями, створений для обміну інформацією з одного комп'ютера на інший за допомогою XML. Протокол SOAP стандартизований за допомогою вбудованих правил і надсилає повідомлення за допомогою інших протоколів, таких як SMTP і HTTP. SOAP часто використовується разом з мовою опису веб-сервісів WSDL, оскільки вона забезпечує спосіб опису інтерфейсу веб-сервісу, включаючи місцезнаходження сервісу і методи, які він підтримує. Це полегшує клієнтам використання веб-сервісу і розуміння того, як з ним взаємодіяти[4].

Спочатку Microsoft розробила цей протокол через проблеми, пов'язані зі старими технологіями, такими як CORBA і DCOM, які не найкраще підходять для обміну інформацією через Інтернет. За допомогою SOAP кілька додатків, створених на різних платформах, що використовують різні мови, можуть спілкуватися один з одним. SOAP має три ключові особливості: розширюваність, тобто ви можете вибирати, які стандартні протоколи ви хочете використовувати; нейтральність, тобто SOAP є нейтральним і може працювати через HTTP, FTP, SMTP і так далі; і незалежність, тобто SOAP не залежить від мови програмування і платформи.

Деякі переваги використання SOAP полягають у тому, що він має вищий рівень безпеки, ніж REST, завдяки WS-Security і ACID, HTTPS і SSL. Ця безпека корпоративного рівня робить його кращим вибором для передачі конфіденційної інформації. .[2]

Що стосується протоколу SOAP, то деякі з його переваг полягають у тому, що він не залежить від мови та платформи, є стандартизованим, забезпечує розширюваність за допомогою стандартів WS, сумісний з ACID та включає обробку помилок.

SOAP дає контроль над наскрізним процесом, таким як комунікація між серверами для внутрішнього використання в організації. Він підтримує як операції зі станом, так і без нього, тоді як REST для простоти розроблений як бездержавний. Крім того, при правильному налаштуванні він є більш безпечним, оскільки використовує захист WS на додаток до підтримки HTTPS і SSL.

SOAP пропонує більшу гнучкість при виборі протоколів передачі даних для обробки різних сценаріїв, таких як HTTP/HTTPS, TCP, FTP, UDP, SMPT і так далі, в той час як REST повинен використовувати HTTP/HTTPS. SOAP також має вбудовану підтримку ACID, яка захищає цілісність бази даних і робить його кращим для конфіденційних транзакцій. Що стосується обробки помилок, то REST не має стандартної системи повідомлень і покладається на повторні спроби зв'язку, в той час як вбудована логіка успішних повторних спроб SOAP забезпечує наскрізну надійність[5].

Якщо вашим пріоритетом є стандартизація, посилена безпека, більша гнучкість протоколів передачі даних, і вас не турбує розмір повідомлень, SOAP буде правильним вибором.

2. REST API

REST — це не протокол, такий як SOAP, а архітектурний стиль з набором принципів для розподілених гіпермедійних систем. REST розшифровується як Representational State Transfer і використовує протокол HTTP для обміну даними. Коли розробники створюють веб-сервіси з використанням REST, їх називають RESTful веб-сервісами.

Керівні принципи REST — це не суворі процедури, а рекомендації, які розробники можуть реалізувати відповідно до своїх вимог. Це робить REST легшим і покращує продуктивність веб-сервісів за рахунок більш швидкого завантаження сторінок, що робить його популярним рішенням для мобільних додатків. Ще одна суттєва відмінність полягає в тому, що REST не обмежується використанням XML, а дозволяє використовувати інші формати повідомлень, такі як JSON, HTML і звичайний текст[6].

Щоб створити REST API, потрібно дотримуватися шести архітектурних обмежень:

- Це вимагає єдиного інтерфейсу
- Клієнт і сервер повинні бути незалежними
- Має бути бездержавним, при цьому кожен запит повинен мати всю необхідну серверу інформацію
- Повинні мати кешовані ресурси
- Вимагає багаторівневої системи
- За необхідності повинна мати виконуваний код

REST був створений для подолання обмежень SOAP за допомогою більш гнучкої архітектури. У порівнянні з SOAP, REST є більш гнучким. Деякі переваги полягають у тому, що він має прості для розуміння стандарти, його легше вивчати і використовувати, він швидкий і ефективний, використовує широко прийнятий стандарт HTTP і має менші формати повідомлень, такі як JSON. З іншого боку, REST не такий незалежний, як SOAP, через необхідність використання кодів стану HTTP.

Основні переваги REST

- Дизайн SOAP складніший і не такий легкий і швидкий, як REST, який вимагає менше ресурсів. Будучи більш легким і гнучким, REST може обробляти передачу повідомлень на багато пристроїв з невеликим обсягом

пам'яті і низькою обчислювальною потужністю, підключених до більш ніж одного сервісу[7].

- Завдяки протоколам HTTP, REST набагато простіший, ніж SOAP, і сумісний з браузерами завдяки формату даних JSON. Можливість працювати не лише з JSON, але й з іншими форматами, такими як XML, HTML та іншими, робить його дуже гнучким при структуруванні повідомлень, форматів, а також клієнтського та серверного масштабу. Архітектура REST базується на принципі клієнт-сервер, що робить користувацький інтерфейс більш портативним, оскільки він відділений від завдань зберігання даних. Крім того, всі компоненти використовують єдиний уніфікований інтерфейс, що спрощує взаємодію додатків з API.
- Кешування HTTP можливе лише на REST, кешування SOAP вимагатиме налаштування додаткового модуля кешування. Це дозволяє зберігати нединамічну інформацію на стороні клієнта, покращуючи швидкість завантаження. REST API, які працюють з JSON-файлами, не потребують накладних блоків тексту і коду, що містяться в XML SOAP, що значно зменшує їх розмір і робить процеси і передачі даних простішими і швидшими. Для пояснення цього часто використовують просту аналогію: SOAP - це «конверт», а REST — «листівка». Листівки (REST) легше читати і вони споживають менше паперу, що означає меншу пропускну здатність; SOAP використовує формат повідомлень у вигляді конверта, і його потрібно «розгорнути», щоб побачити повідомлення всередині, таким чином збільшуючи використання пропускної здатності.[8]

1.3. Актуальність теми та пошук аналогів

У процесі дослідження наявних рішень для відстеження повітряних тривог на територіях України за допомогою Discord API не було знайдено

аналогів програмного забезпечення з подібними функціями. Це свідчить про унікальність та актуальність створення такого програмного продукту.

Незважаючи на це зауваження варто відзначити те, що Discord є популярною платформою для розробки ботів з широким спектром функцій для використання у різних галузях — від розваг до навчання і управління спільнотами та бізнес-процесами і це свідчить про збільшений інтерес до автоматизація процесів та інтеграція інструментарію у цифровому середовище.

1.1.1. Бот Soccer Guru

Soccer Guru — інтерактивний бот для шанувальників футболу. Цей бот демонструє можливості платформи Discord у наданні користувачам тематичного досвіду через гейміфікацію, інтеграцію ігор та взаємодію спільноти[11].

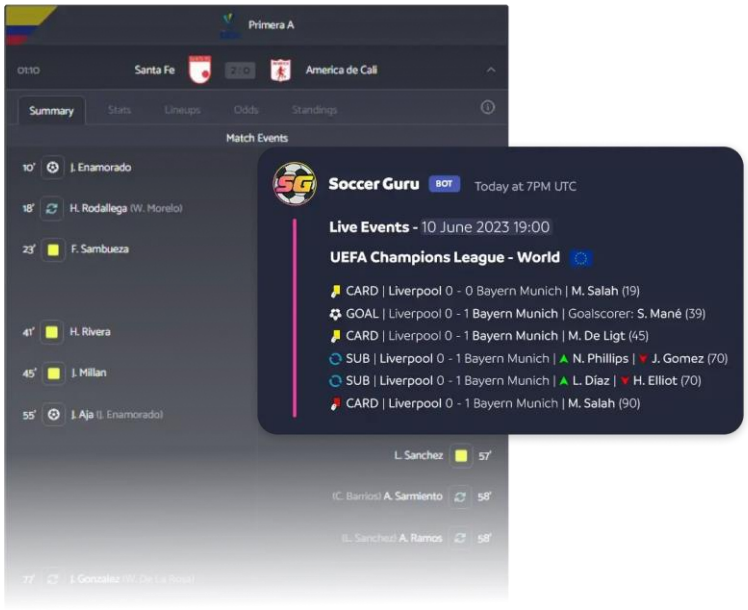


Рисунок 1.4. — Інтерфейс боту Soccer Guru

Однією з ключових особливостей Soccer Guru є можливість створення власних віртуальних футбольних клубів. Ці клуби діють як самодостатні групи: гравці формують команди, беруть участь у змаганнях та прагнуть стати

переможцями у віртуальних лігах. Результати визначаються сумою ігрової статистики та невпливовими факторами, що додає елемент випадковості та несподіваності.

Крім того, Soccer Guru пропонує розширену економіку з можливістю купувати та продавати віртуальних гравців. Ця система надає користувачам можливість поліпшити свою команду й боротися з іншими учасниками більш ефективно. Такий підхід робить процес формування команди не лише грою на удачу, але й стратегічним завданням.

Цікавою можливістю бота є також можливість взаємодіяти з між-серверними змаганнями, що дозволяють користувачам різних серверів спілкуватися між собою через участь в загальних турнірах. Такий вид активності значно покращує взаємодію між серверами, що збільшує популярність самого боту та залучає нових користувачів до серверів.

На даний момент бот використовують **190 тисяч** серверів.

1.1.2. Бот iTranslator

iTranslate — це бот для платформи Discord, який надає можливість миттєво перекладати повідомленя між різними мовами з метою полегшення комунікацію між користувачами з різних країн, що робить його важливим інструментом для міжнародних спільнот.

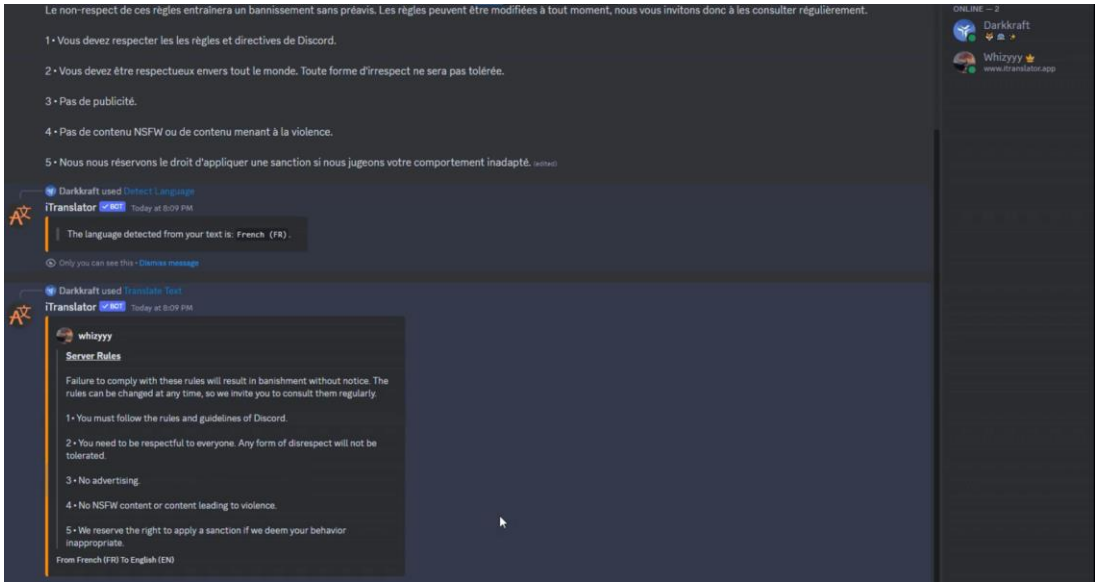


Рисунок 1.5. — Інтерфейс боту iTranslate

Функціональні можливості додатку iTranslate

Однією з важливих можливостей бота є автоматичний переклад текстових повідомлень. Користувачі можуть налаштовувати бот для перекладу як окремих повідомлень, так і всіх текстів у каналі для повноцінного використання перекладу під час спілкування між учасниками сервера на різних мовах зі збереженням розуміння один одного.

Бот підтримувати більше 100 мов — включаючи такі популярні як англійська та іспанська, а також менш поширені мови як українська чи ісландська. Використовуючи сучасні алгоритми перекладу дозволяють досягти високо точного та природного перекладу навіть у складних конструкціях.

iTranslate також має взаємодію через команди інтерфейсу для користувачів, вони можуть отримати швидкий переклад окремого тексту за допомогою вказівки мови оригіналу та мови перекладу. Це зручне рішення для обробки одноразових запитів на переклад, яке не потребує зміни налаштувань каналу[11].

На даний момент бот використовують 60 тисяч серверів.

1.1.3. Бот NotifyMe

NotifyMe — це універсальний бот для Discord, призначений для відслідковування та сповіщення користувачів про різноманітні події. Його основна мета полягає в автоматизацію отримання важливих повідомлень, що робить його корисним для персональних та колективних потреб серверами.

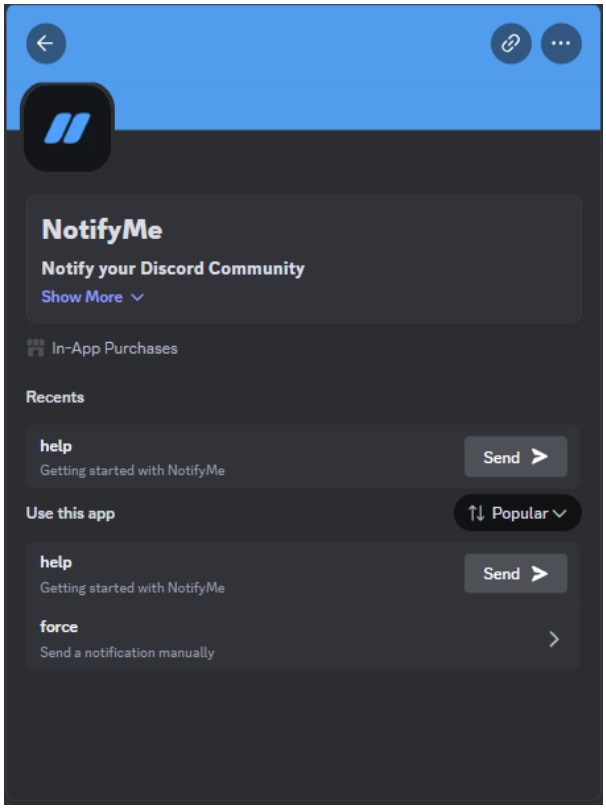


Рисунок 1.6. — Інтерфейс бота NotifyMe

Він дозволяє користувачам налаштовувати індивідуалізовані сповіщення. Цей бот підтримуватиме широкий спектр джерел для отримання інформацію: RSS-стрічки, соцмережі, веб-сайти та внутрішні повідомлення з інших каналів Discord.

Одній з важливих функцій бота є можливість інтеграцію з API різних сервісів для отримання актуальних даних у реальному часі. Наприклад, користувачам надаються можливості отримувати сповіщення про новини, оновлення погоди або зміни на фінансових ринках або навіть подіях у

геймерських світах. Завдяки цьому бот може використовуватися як універсальний інформаційний центр.

NotifyMe став дуже популярним інструментом для серверів з великою кількістю користувачів, а також для особистого використання. На даний момент бот використовують 73 тисячі серверів[12].

Висновки до першого розділу .

У першому розділі було розглянуто основні аспекти, пов’язані з розробкою Discord бота. Зокрема, описано платформу Discord, її основні можливості та функціонал, який відкривається для користувачів через боти. Завдяки детальному аналізу глосарію Discord було зрозуміло, як різноманітні функції, такі як текстові та голосові канали, взаємодіють із ботами та що вони можуть робити в контексті сповіщень.

Наступним етапом був аналіз API та їх ролі в роботі з ботами. Було розглянуто, що таке API, як вони функціонують і яку роль вони відіграють у інтеграції з зовнішніми сервісами. Порівнявши два основні типи API — SOAP та REST, обрано REST, оскільки він є більш гнучким та зручним для розробки бота, що працюватиме з постійно змінюваними даними.

Актуальність розробки бота для сповіщення про повітряні тривоги була підтверджена тим, що подібний проект не тільки відповідає на потребу в інформації, але й має соціальне значення. Вивчено існуючі успішні Discord боти, такі як Soccer Guru, iTranslator та NotifyMe, що демонструють, як можна ефективно інтегрувати різні функції сповіщень та автоматизації в Discord, а також забезпечити корисний та інтуїтивно зрозумілий досвід для користувачів. Ці приклади підтвердили важливість проекту і дали натхнення для подальшої розробки.

Отже, розробка бота для Discord з використанням REST API є доцільною та перспективною ідеєю, яка дає можливість створити соціально корисного бота, який буде не лише корисним, а й допоможе врятувати життя.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

2.1. Визначення вимог до системи моніторингу повітряних тривог у Discord

Система моніторингу повітряних тривог повинна надійно та швидко виявляти оновлення про повітряні тривоги з авторитетного джерела, інтегруючись з відповідним API для отримання актуальною інформацію про тривоги у реальному часі.

Другою важливим вимогою є можливість налаштування параметрів моніторингу для кожного Discord-сервера. Програма повинна запам'ятовувати географічні області, які цікавлять користувачів і надсилати повідомлення в конкретні текстові та голосові канали відповідно до цих налаштувань.

Важливо мати зручний та зрозумілий інтерфейс для користувачів. Наприклад, системний адміністратор повинен мати можливість легко налаштовувати списки регіонів та формати повідомлень і обирати канали для надсилання сповіщень.

Основною можливістю системи є автоматичне отримання інформації про тривоги, їх обробка та відправлення сповіщень через текстові та голосові канали. Щоб реалізувати цю функціональну можливість використовуватиметься надійний API для оперативного доступу до даних про повітряні тривоги.

Крім того, програма повинна мати можливість працювати в різних каналах на платформі Discord і враховувати обмеження цього середовища роботи такі як швидкість відправки повідомлень та обмеження на запити до API.

Ці вимоги сприятимуть ефективній роботі системи моніторингу повітряних тривог у Discord та забезпечать зручне використання для адміністраторів серверів і користувачів.

У цьому розділі будуть зазначені інструменти та середовище розробки для створення програмного забезпечення. Також буде детально розглянуто

використання відкритого API для отримання даних про повітряні тривоги: формат переданих даних та принципи його функціонування будуть описані усередині цього розділу для полегшення інтеграцію API у вашу розробку та правильною обробкою отриманих даних.

2.2. Вибір засобів розробки

Під час розробки програмного забезпечення для моніторингу повітряних тривог у Discord важливим етапом є вибір правильних інструментів розробки. При аналізі особливостей взаємодій з API та інтеграцію з платформою Discord та перегляді доступних мов програмування було визначено, що найбільш практичним варіантом для реалізації ї цього проекту є мова програмування C#[14].

C# добре підходить для взаємодії з платформою Discord завдяки наявності бібліотек для інтеграцій — наприклад DSharpPlus чи Discord.Net — які надають готові розв'язки для обробки подій та взаємодії з API Discord-у та керування ботом, що суттєво прискорює процес розробки[15].

C# — це сучасна мова програмування, створена компанією Microsoft у 2000 році в рамках платформи .NET. Вона була розроблена як універсальна мова для створення широкого спектра програмних рішень: від програм і мобільних застосунків до веб-додатків та ігор. Завдяки своїй простоті, продуктивності та широким можливостям, C# став одним із найпопулярніших інструментів для розробників у всьому світі.

C# є об’єктно-орієнтованою мовою програмування. Це означає, що розробка програм на C# базується на принципах об’єктно-орієнтованого програмування (ООП). ООП дозволяє структурувати код у вигляді об’єктів, які моделюють реальні чи абстрактні сутності, забезпечуючи їхні властивості та методи.

Об'єктно-орієнтоване програмування (ООП) у C# — це потужна парадигма, яка структурує код навколо об'єктів, інкапсулює дані та поведінку.

Вона використовує успадкування, поліморфізм та абстракції для багаторазового використання та модульності. ООП сприяє створенню чистих, легко підтримуваних і масштабованих додатків, підвищуючи ефективність і структуру розробки програмного забезпечення. У C# класи виступають в ролі шаблонів для об'єктів, що дозволяє створювати безліч екземплярів з унікальними атрибутами. Інкапсуляція забезпечує безпеку даних, приховуючи деталі реалізації. Спадкування полегшує повторне використання коду, а поліморфізм дозволяє об'єктам набувати різних форм, підвищуючи гнучкість. Абстрагування спрощує складні системи, роблячи їх легшими для розуміння та модифікації. ООП в C# сприяє систематичному та інтуїтивно зрозумілому підходу до розробки програмного забезпечення.

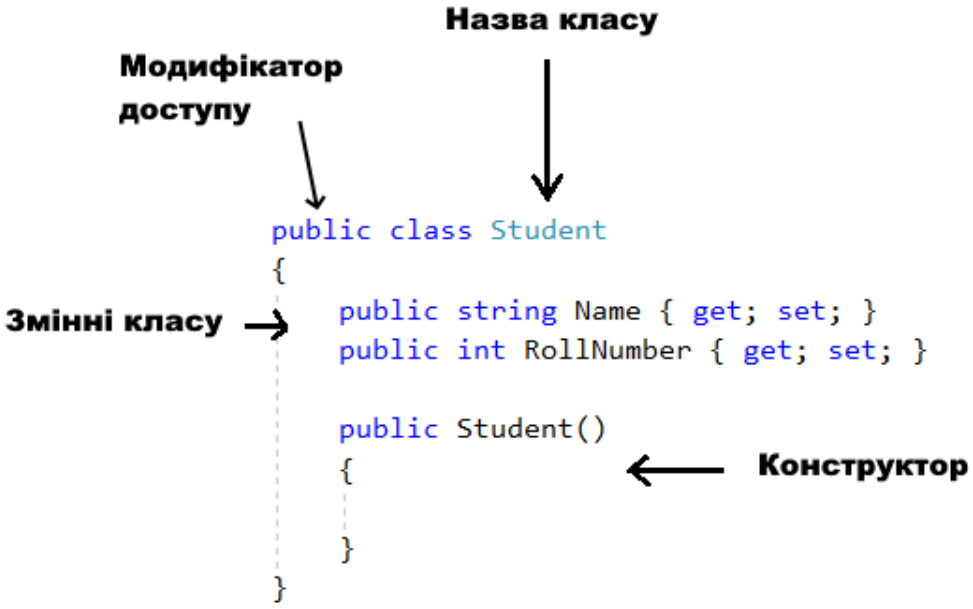


Рисунок 2.1. — Структура класу в мові програмування C#

В об'єктно-орієнтованому програмуванні (ООП) об'єкти є екземплярами класів і представляють сутності або концепції реального світу. Вони інкапсулюють як дані (властивості), так і поведінку (методи) в єдине ціле.

Об'єкти взаємодіють один з одним, спілкуються за допомогою методів або повідомлень і мають унікальну ідентичність та стан.

Давайте розглянемо реальне застосування об'єктів в ООП. У реальному світі тварину можна представити як об'єкт в ООП. Припустимо, у нас є клас під назвою «Animal», який визначає загальні атрибути та поведінку тварин[16].

Атрибутами об'єкта «Animal» є тип (наприклад, собака, кіт, птах), колір, вік, звук. Поведінкою об'єкта «Animal» може бути їсти, спати, видавати звуки або рухатися.

У цьому прикладі клас «Animal» виступає в ролі шаблону, визначаючи загальні властивості та поведінку всіх тварин. Кожна конкретна тварина, наприклад, собака, кіт або птах, є об'єктом, що представляє унікальний екземпляр класу " Animal" з його специфічними атрибутами та поведінкою.

Вони сприяють повторному використанню коду і забезпечують модульний та інтуїтивно зрозумілий спосіб представлення та маніпулювання складними системами.

Вибір C# для розробки проекту обумовлений його потужними можливостями, зокрема інтеграцією з платформою Discord. C# підтримує бібліотеки, такі як *DSharpPlus* або *Discord.Net*, які спрощують розробку ботів і інтеграцію з API Discord. Крім того, C# має багатий набір функцій для роботи з даними, включаючи обробку JSON та асинхронне програмування, що є важливим для швидкої обробки даних про повітряні тривоги.

До переваг мови C# також належать інтуїтивно зрозумілий синтаксис, підтримка різних платформ, висока продуктивність та багата екосистема інструментів і бібліотек. Visual Studio, як основне середовище розробки, забезпечує ефективну підтримку написання, тестування та відлагодженням коду. Завдяки всім цим характеристикам, C# є ідеальним вибором для реалізації програмного забезпечення моніторингу повітряних тривог у Discord.

Абстракція в C#. Абстрагування — це фундаментальний принцип об'єктно-орієнтованого програмування (ООП), який дозволяє розробникам створювати ефективний, легко підтримуваний і масштабований код. Він передбачає зосередження уваги на основних функціях, приховуючи непотрібні деталі. У C# абстракція досягається завдяки використанню абстрактних класів, інтерфейсів та абстрактних методів.

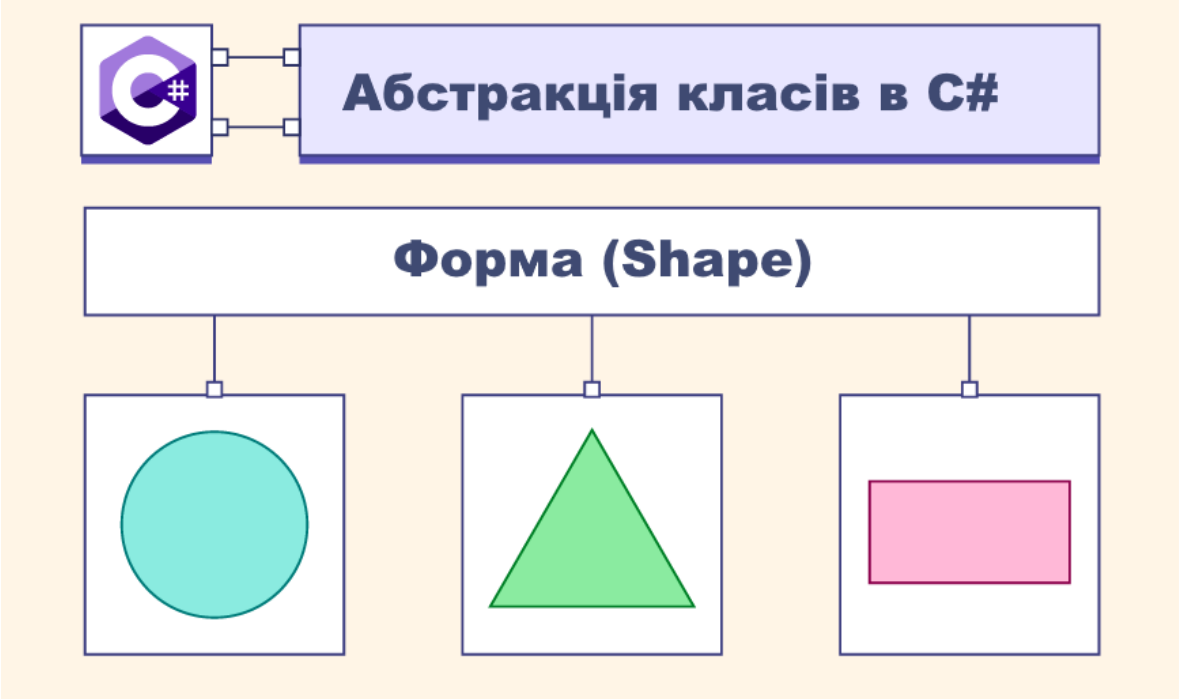


Рисунок 2.2. — Абстракція класів в мові програмування C#

По суті, абстракція дозволяє створювати моделі, які представляють реальні об'єкти або системи. Ці моделі відображають основні характеристики та поведінку об'єктів, з якими ми працюємо, абстрагуючись від деталей реалізації. Таким чином, абстракція забезпечує високорівневу перспективу, дозволяючи нам зосередитися на «що», а не на «як». Це сприяє модульності коду, повторному використанню та гнучкості[17].

Поліморфізм в C#. Поліморфізм — це ключова концепція об'єктно-орієнтованого програмування (ООП), яка дозволяє розглядати об'єкти різних класів як об'єкти спільного базового класу або інтерфейсу. Це дозволяє використовувати один і той самий код з різними типами об'єктів, сприяючи гнучкості, повторному використанню коду та розширенню.

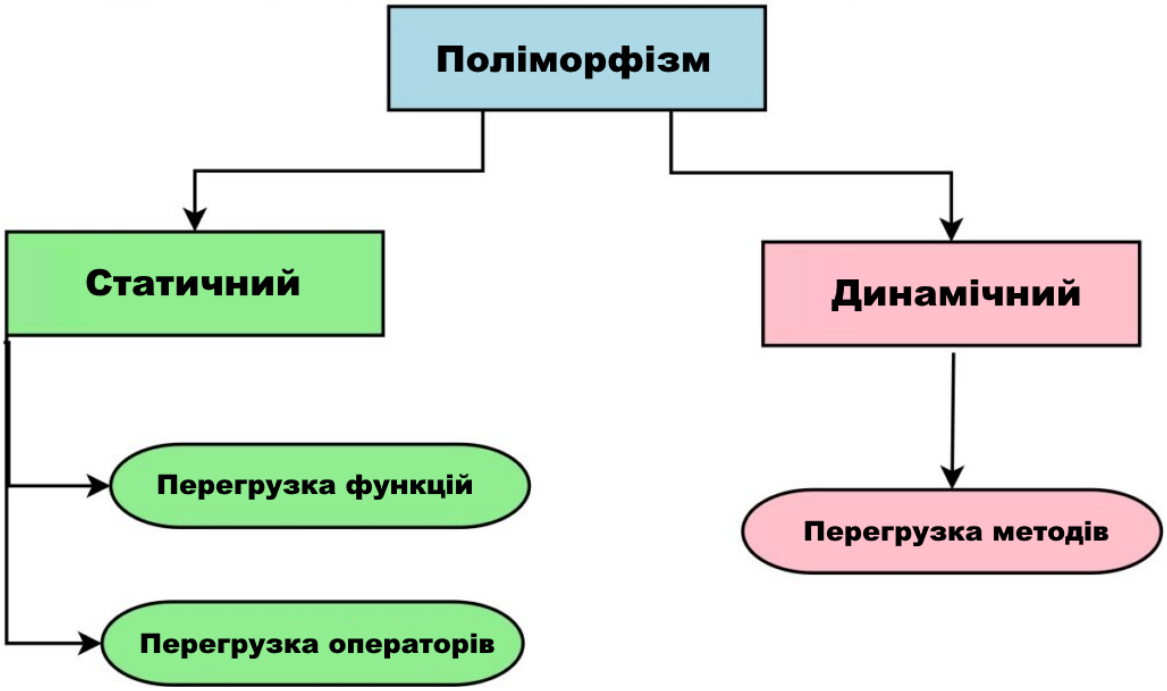


Рисунок 2.3. — Типи поліморфізму та їх функціональність

Поліморфізм походить від грецьких слів «полі» (що означає багато) і «морф» (що означає форма). В контексті ООП поліморфізм дозволяє розглядати об'єкти різних класів як об'єкти спільного базового класу або інтерфейсу. Це означає, що ми можемо використовувати один інтерфейс коду для представлення декількох типів об'єктів.

У C# поліморфізм можна розділити на два типи: статичний поліморфізм та динамічний поліморфізм.

- Статичний поліморфізм досягається за рахунок перевантаження методів та операторів. Це дозволяє викликати різні методи або оператори в залежності від кількості, типу або порядку аргументів під час компіляції.
- Динамічний поліморфізм досягається через перевизначення методів та інтерфейсів. Це дозволяє об'єктам різних похідних класів розглядатися як об'єкти базового класу або інтерфейсу, а відповідний метод визначається під час виконання на основі фактичного типу об'єкта.

Інкапсуляція в C#. Інкапсуляція — це один з фундаментальних принципів об'єктно-орієнтованого програмування (ООП), який просуває концепцію об'єднання пов'язаних даних і поведінки в межах класу. Він інкапсулює внутрішній стан (дані) об'єкта і забезпечує контрольований доступ до цього стану ззовні класу.

Властивості інкапсуляції в C#. Властивості надають спосіб інкапсуляції приватних полів та розкриття їх через контрольовані точки доступу. Вони дозволяють нам визначати власну логіку для читання і запису даних, забезпечуючи цілісність даних і дотримання бізнес-правил.

У C# властивості визначаються за допомогою аксесорів get та set.

Аксесор get отримує значення властивості, а аксесор set встановлює значення властивості. Ми також можемо вказати різні модифікатори доступу для аксесорів get і set, щоб керувати доступом на читання і запис окремо[18].

2.3. Вибір середовища та інструментів розробки

2.3.1. Microsoft Visual Studio

Visual Studio — це програмне забезпечення для розробників, яке створила компанія Microsoft і відоме як один з найсильніших інструментів для професіоналів у сфері розробки програмного забезпечення. Воно підтримує різноманітні мови програмування - таких як C#, C++, Python і JavaScript — що робить його універсальним рішенням для створення широкого спектру застосунків: веб та мобільних додатків до складних корпоративних систем.

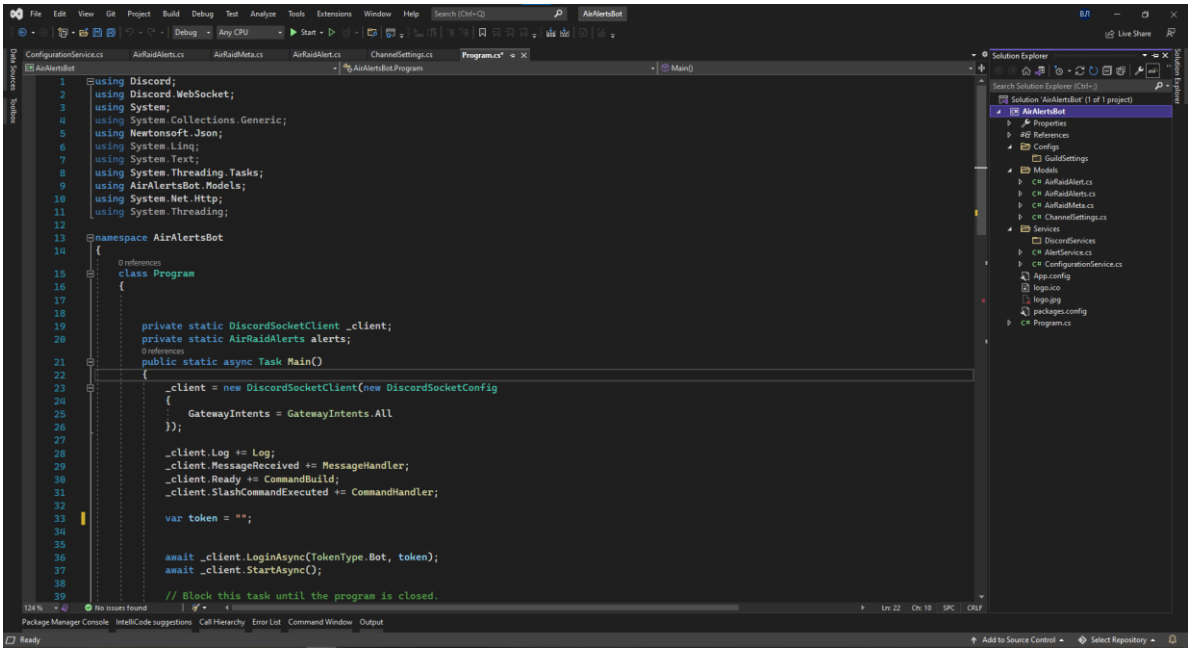


Рисунок 2.4. — Інтерфейс Visual Studio

Однією з важливих функцій Visual Studio є її інтеграція з платформою .NET для продуктивності розробки на C#. Ця інтеграція дозволяє удосконалити швидкість розробки завдяки розширеному автозавершенню коду та вбудованим інструментам для покращення коду для утримання чистоти та зрозумілості коду.

Visual Studio маючи надійний інтерфейс для налагодження коду дозволяє розробникам виконувати його крок за кроком і контролювати змінні та обробляти винятки що особливо важливо при створенні складних проектів наприклад бот Discord де розуміння взаємодії між компонентами системи є критично важливим.

Одній з важливих переваг є вбудовані інструменти Git, які дозволяють розробникам ефективно управляти версіями коду, співпрацювати в команді та мають детальну історію змін у проекті. Крім цього, Visual Studio підтримує плагіни та розширення для налаштування середовища роботи згідно з конкретними потребами кожного проекту.

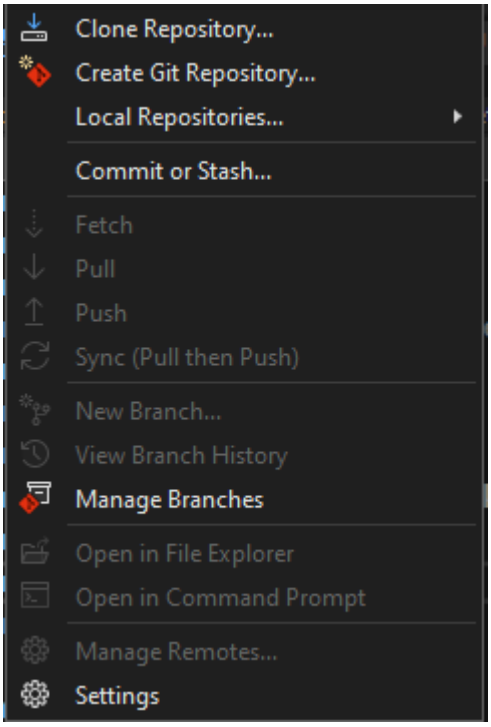


Рисунок 2.5. — Функціональні можливості інструменту Git

IDE також добре працює з асинхронним програмуванням і надає розробникам інструменти для тестування та налагодження інтеграції API та асинхронних операцій. Це є критично важливим для ефективності при роботі з API повітряних тривог чи подібними системами.

Отже Visual Studio — це незамінний інструмент для програмістів з високою продуктивністю та гнучкість у роботі з проектами будь-якої складності.

2.3.2. Сервіс alerts.in.ua

Сервіс alerts.in.ua що пропонує широкий доступ API для відстеження попереджень про небезпеку у повітрі по всій території України. Ця платформа збирає дані в режимі реального часу з офіційних місцевих джерел і представляє їх в структурованому форматі, який можна легко інтегрувати в різні програми.

API призначено для швидкої передачі інформації для того щоб користувачі могли бути в курсі статусу попереджень про повітряну небезпеку у всіх регіонах. Цей API підтримує широкий спектр параметрів запитань, що дає розробникам можливість отримувати дані, адаптовані до їх потреб, наприклад, інформація про конкретні регіони.

Зосереджуючись на надійності та легкості використання, сервіс alerts.in.ua має ключову роль у створенні можливостей для розробників у сфері сповіщень, аналізу та візуалізація тривог. Простий процес впровадження та зрозуміла документація роблять його оптимальним вибором для проектів з моніторингу попереджень щодо повітряних небезпек в України у реальному часу.

Усі методи API можуть повертати помилки. Помилки повертаються в форматі JSON.

Таблиця 2.1. — Коди відповедей API alerts.in.ua

Код	**Назва**	**Опис**
200	OK	Запит виконано успішно

Код	Назва	Опис
304	Не змінено	Дані не було змінено
401	Не авторизовано	Токен API відсутній, неправильний, відкликаний або прострочений.
403	Заборонено	Ваш IP адрес заблокований або API не доступне в вашій країні.
429	Забагато запитів	Ліміт запитів на хвилину перевищено

В ході роботи API може повертати location_uid, що являє собою заздалегідь визначений унікальний ідентифікатор міста, або області. Нижче в таблиці 2.2 зазначено кожен location_uid та локацію, за яку він відповідає[19].

Таблиця 2.2. — Унікальні ідентифікатори міст та областей

UID	Назва області/міста
3	Хмельницька область
4	Вінницька область
5	Рівненська область
8	Волинська область
9	Дніпропетровська область

UID	Назва області/міста
19	Полтавська область
20	Сумська область
21	Тернопільська область
22	Харківська область
23	Херсонська область

UID	Назва області/міста
10	Житомирська область
11	Закарпатська область
12	Запорізька область
13	Івано-Франківська область
14	Київська область
15	Кіровоградська область
16	Луганська область
17	Миколаївська область
18	Одеська область

UID	Назва області/міста
24	Черкаська область
25	Чернігівська область
26	Чернівецька область
27	Львівська область
28	Донецька область
29	Автономна Республіка Крим
30	м. Севастополь
31	м. Київ

Розглянемо основні типи запитів alerts.in.ua:

- **Статус повітряних тривог в областях**

/v1/iot/active_air_raid_alerts_by_oblast.json

Повертає стан повітряних тривог в областях. Компактне API для використання в IoT пристроях.

Результат повертається у вигляді JSON, що містить рядок:

"ANNNNNNNNNNANNNNNNNNNNNN"

В якому кожна літера відповідає за свій вид тривоги в певній області, пояснення наведене в таблиці 2.3.

Таблиця 2.3. — Таблиця скорочень виду тривоги

Код	**Значення**
A	Повітряна тривога активна в усій області
P	Часткова тривога в районах чи громадах
N	Немає інформації про повітряну тривогу

Для кожної букви рядка є своя область в наступному порядку:

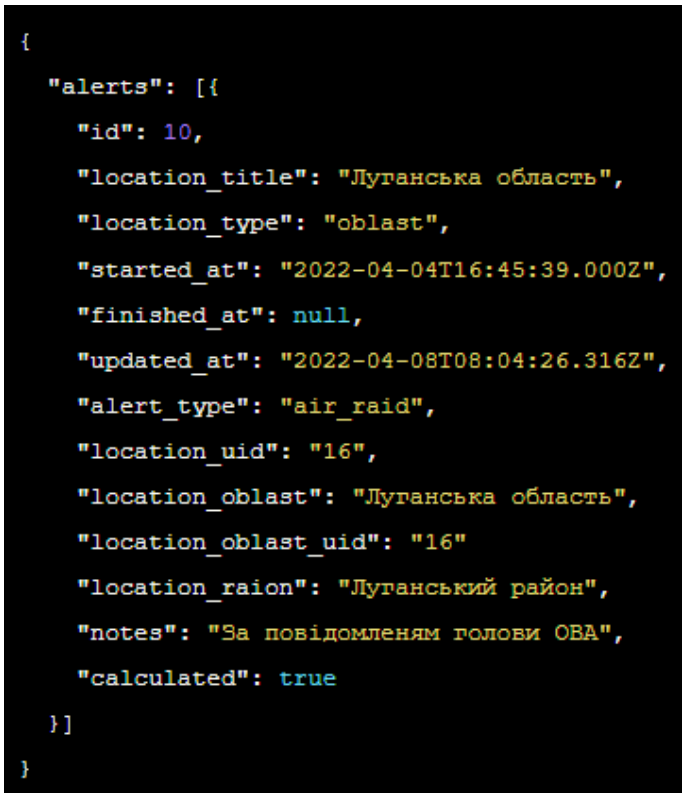
"Автономна Республіка Крим", "Волинська область", "Вінницька область", "Дніпропетровська область", "Донецька область", "Житомирська область", "Закарпатська область", "Запорізька область", "Івано-Франківська область", "м. Київ", "Київська область", "Кіровоградська область", "Луганська область", "Львівська область", "Миколаївська область", "Одеська область", "Полтавська область", "Рівненська область", "м. Севастополь", "Сумська область", "Тернопільська область", "Харківська область", "Херсонська область", "Хмельницька область", "Черкаська область", "Чернівецька область", "Чернігівська область"

Тобто перша буква в рядку — статус повітряної тривоги в Автономній Республіці Крим, друга — в Волинській області, третя — в Вінницькій області і так далі.

- **Список активних тривог**

/v1/alerts/active

Повертає список регіонів в яких активна повітряна тривога чи будь-яка інша загроза у форматі зображеному на рисунку 2.6.



```
{
  "alerts": [{
    "id": 10,
    "location_title": "Луганська область",
    "location_type": "oblast",
    "started_at": "2022-04-04T16:45:39.000Z",
    "finished_at": null,
    "updated_at": "2022-04-08T08:04:26.316Z",
    "alert_type": "air_raid",
    "location_uid": "16",
    "location_oblast": "Луганська область",
    "location_oblast_uid": "16"
    "location_raion": "Луганський район",
    "notes": "За повідомленям голови ОВА",
    "calculated": true
  }]
}
```

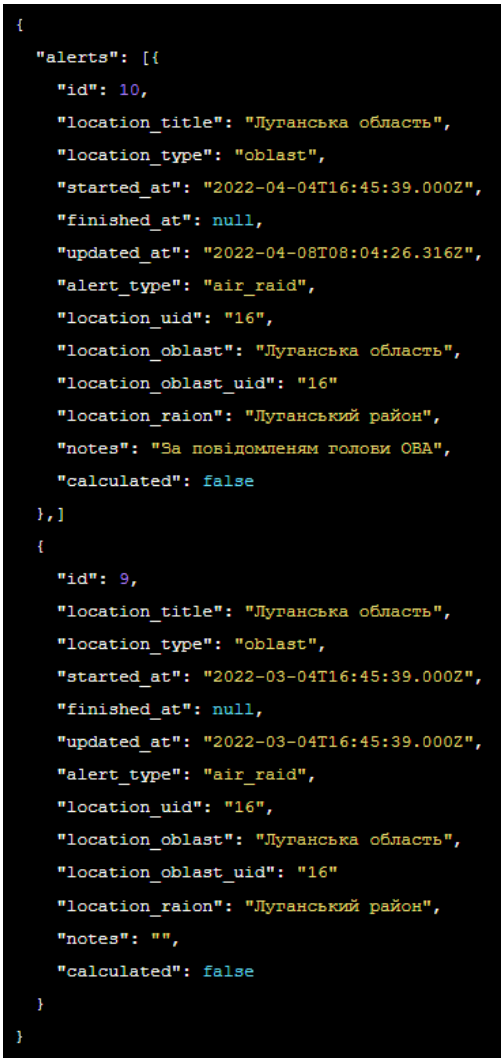

Рисунок 2.6. — Формат даних всіх активних тривог

- **Повертає історію тривог за певний період**
  /v1/regions/:uid/alerts/:period.json

Цей запит API має два параметри:

- **uid** — Унікальний ідентифікатор області
- **period** — Період для якого повертається історія тривог.

Повертає список тривог за вказаний період у форматі зображеному на рисунку 2.7.



```
{
  "alerts": [{
    "id": 10,
    "location_title": "Луганська область",
    "location_type": "oblast",
    "started_at": "2022-04-04T16:45:39.000Z",
    "finished_at": null,
    "updated_at": "2022-04-08T08:04:26.316Z",
    "alert_type": "air_raid",
    "location_uid": "16",
    "location_oblast": "Луганська область",
    "location_oblast_uid": "16"
    "location_raion": "Луганський район",
    "notes": "За повідомленям голови ОВА",
    "calculated": false
  },]
  {
    "id": 9,
    "location_title": "Луганська область",
    "location_type": "oblast",
    "started_at": "2022-03-04T16:45:39.000Z",
    "finished_at": null,
    "updated_at": "2022-03-04T16:45:39.000Z",
    "alert_type": "air_raid",
    "location_uid": "16",
    "location_oblast": "Луганська область",
    "location_oblast_uid": "16"
    "location_raion": "Луганський район",
    "notes": "",
    "calculated": false
  }
}
```


Рисунок 2.7. — Формат даних тривог за вказаний період

2.3.3. JSON та бібліотека Newtonsoft.Json

Якщо коротко, JSON розшифровується як JavaScript Object Notation — це стислий ієрархічний синтаксис серіалізації даних, який підтримується всіма сучасними браузерами.

Його формат робить його легким способом представлення об'єктів, залишаючись при цьому читабельним для людини. З цієї причини він замінив нотацію XML на багатьох платформах.

XML чудово підходить для опису ієрархій об'єктів і навіть семантики, але додає багато накладних витрат для серіалізованого об'єкта[20].

В рисунку 2.8 наведено приклад простого об'єкта User, серіалізованого у XML.

```
<xml>
   <user>
       <firstName>Jason</firstName>
       <middleName>Alexander</middleName>
       <lastName>Smith</lastName>
       <address>
           <street1>1234 Someplace Avenue</street1>
           <street2>Apt. 302</street2>
           <city>Anytown</city>
           <state>NY</state>
           <postalCode>12345</postalCode>
           <country>US</country>
       </address>
   </user>
</xml>
```

Рисунок 2.8. — Приклад об’єкта User в XML

Ті самі дані, представлені в JSON на рисунку 2.9, є набагато ефективнішими, а також більш читабельними для людини.

```
{
  "firstName" : "Jason",
  "middleName" : "Alexander",
  "lastName" : "Smith",
  "address" : {
    "street1" : "1234 Someplace Avenue",
    "street2" : "Apt. 302",
    "city" : "Anytown",
    "state" : "NY",
    "postalCode" : "12345",
    "country" : "US"
  }
}
```

Рисунок 2.9. — Приклад об’єкта User в JSON

JSON є крос-платформним. Оскільки підтримка браузерами дуже поширена, JSON також є кросплатформенним способом представлення даних. У цьому полягає справжня сила JSON. Його можна використовувати безпосередньо в JS без будь-якого синтаксичного аналізу, інтерпретації або сторонніх бібліотек[21].

Основні типи даних в JSON:

- рядок (міститься в лапках)
- число (цілі та десяткові числа)
- булеві (записані як true або false, без лапок)
- об'єкт (обгорнутий фігурними дужками, створюючи ієрархію)
- масив (береться у квадратні дужки, масиви можуть містити списки будь-якого типу, включаючи змішані типи)
- null (пишеться точно так само, як null, для позначення не-значення)

Newtonsoft.Json. Newtonsoft.Json або Json.NET — відома бібліотека з відкритим вихідним кодом для платформи .NET, яка призначена для роботи з даними у форматі JSON (JavaScript Object Notation). Ця бібліотека надає користувачам потужні можливості для перетворення об'єктів у формат JSON та навпаки — десеріалізувати дані JSON у об'єкти .NET Framework. Створена Джеймсом Ньютон-Кінгом ця бібліотека користується широкою популярністю завдяки своїй простоті використання та ефективності[22].

Для початку роботи з бібліотекою Newtonsoft.Json необхідно відкрити інструмент NuGet Package Manager, як зображено на рисунку 2.10.

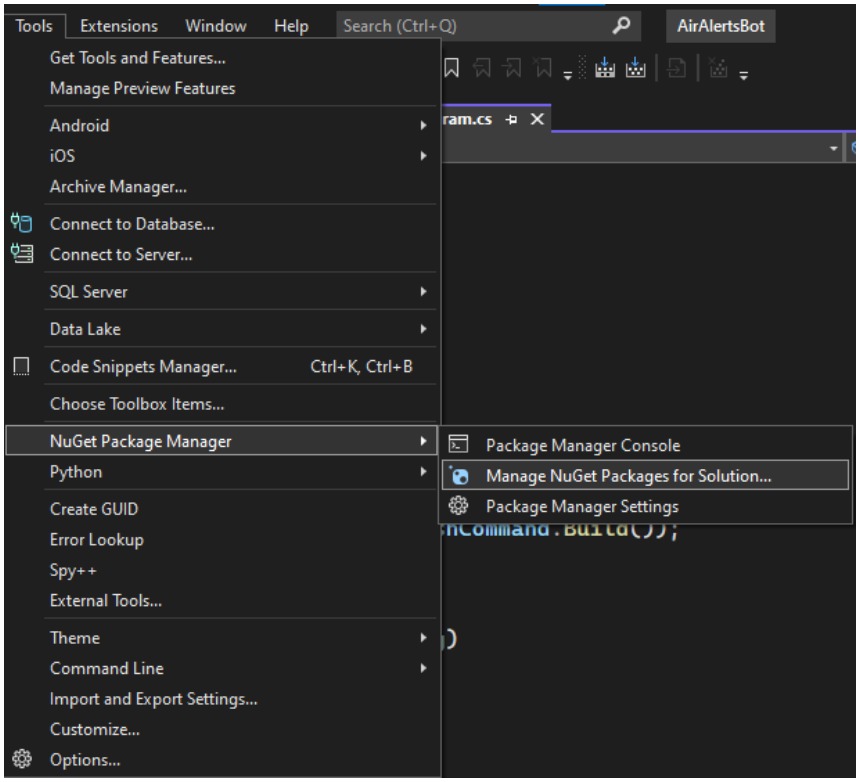


Рисунок 2.10. — інструмент NuGet Package Manager

Далі в пошуку необхідно вказати назву бібліотеки, обрати проект та натиснути Install (встановити), як зображено на рисунку 2.11.

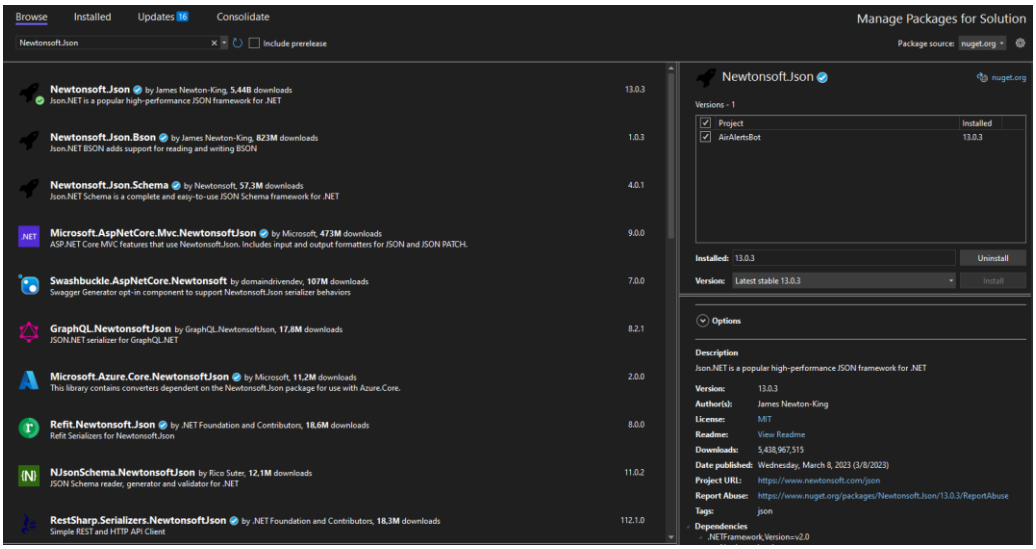


Рисунок 2.11 — Пошук та встановлення бібліотеки

Для того щоб перетворити дані з формату JSON у об’єкти для подальшої обробки потрібно створити спеціальний клас з усіма властивостями відповідними полям у JSON. Коли клас буде готовий до використання ми можемо скористатися методами бібліотеки Newtonsoft.Json для перетворення тексту у форматі JSON і заповнення нового об’єкту нашого класу даними з файлу.

Також можна конвертувати будь□який об’єкт .NET у JSON рядок. Це дуже зручно для передачі даних до API або для збереження у файлі. Бібліотека гарантуватиме правильний формат JSON і включить всю необхідну інформацію з об’єкту[23].

Висновки до другого розділу .

У цій частині розглянуто основні вимоги до бота і обрані технологічні рішення для його розробки. Аналіз предметною областю у першому розділі дозволив чітко визначити ключові функціональності та можливості, якими повинен надавати програмний продукт.

Мовою програмування C# було обрано через його головні переваги — високий рівень надійності та зручність у роботі з асинхронними операціями. C# підтримує об'єктно-орієнтоване програмування (ООП), що дозволяє створювати модульні та легкозмінювані системи. Використання принципів ООП — інкапсуляція, наслідування та поліморфізм — сприяє організації коду та забезпечують велику гнучкість при розробці складних систем..

Було обрано середовище Visual Studio для розробки програм на мові C#. Інтегроване середовище розробки володіє різноманітним набором інструментів для написання коду та тестування програмного забезпечення, що полегшує процес розробки значною мірою. Крім цього, у Visual Studio є можливості для роботи з базами даних і контролем версій, що сприяють ефективності та зручності виконання проектів будь-якою складністю.

Служба alerts.in.ua була обрана для отримання інформації про повітряні тривоги. Для обробки даних використано JSON формат та бібліотеку Newtonsoft.Json, яка надає можливість зручно працювати з цим форматом. Ці технології і інструменти є найкращими вибором для розробки чат-бота та гарантують швидку та ефективну реалізацію проекту.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

3.1. Створення та налаштування боту в Discord

Процес створення нового Discord-бота розпочинається із входу на офіційний Discord Developer Portal (рис.3.1), де розробники можуть керувати своїми додатками та створювати нових ботів.

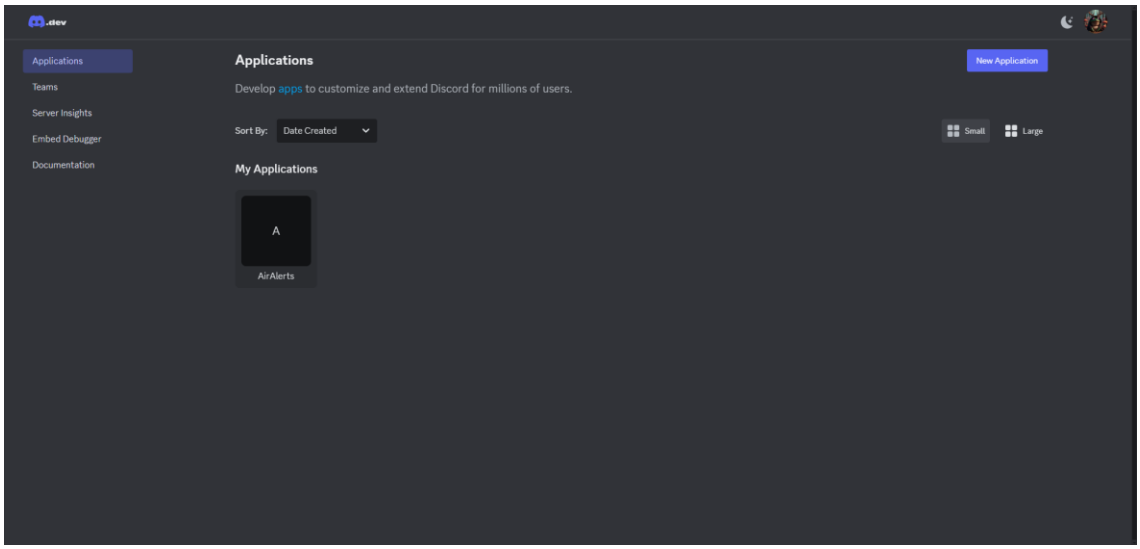


Рисунок 3.1. — Головне меню порталу Discord Developer

Після авторизації потрібно натиснути кнопку "New Application". У вікні, що з'явиться, слід ввести назву для майбутнього бота. Ця назва буде видимою для користувачів та слугуватиме ідентифікатором вашого бота. На цьому етапі можна також завантажити логотип для бота, щоб зробити його більш впізнаваним.

Після створення додатка ви перейдете на сторінку налаштувань (рис.3.2). Тут містяться основні дані про вашого бота, такі як його ідентифікатор (ID) і доступ до розширених налаштувань.

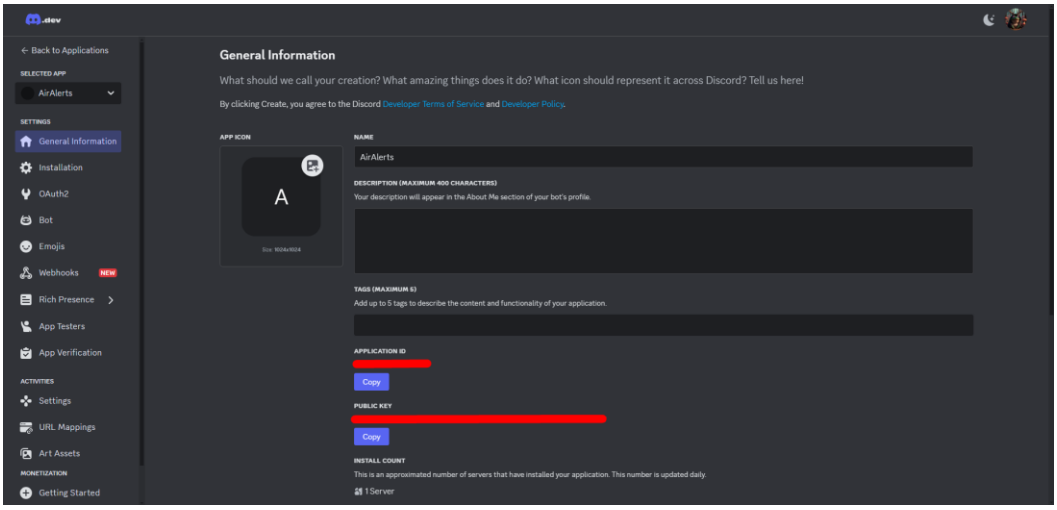


Рисунок 3.2. — Сторінка для налаштування додатку

Наступним кроком є перехід на вкладку "Bot" (рис.3.3), де необхідно натиснути кнопку "Add Bot", щоб створити бота як окрему сутність. Після цього автоматично генерується Token, який є ключовим компонентом для взаємодії з Discord API. Цей токен потрібно зберегти в безпечному місці, оскільки він забезпечує доступ до управління ботом. Якщо токен стане доступним стороннім особам, зловмисники зможуть використовувати вашого бота для власних цілей.

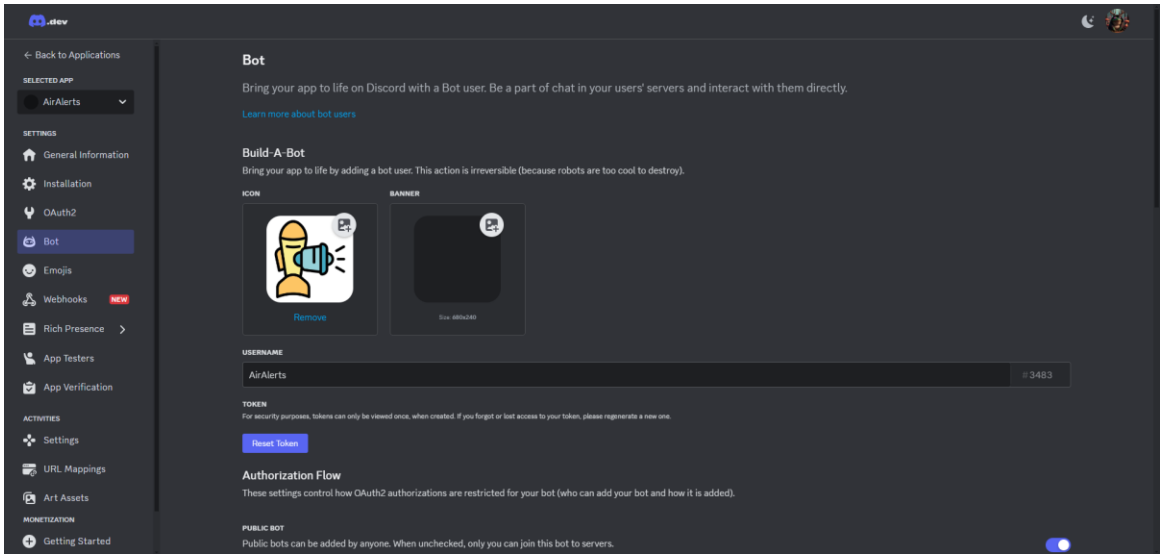


Рисунок 3.3. — Сторінка для налаштування бота

На вкладці налаштувань бота (рис.3.4) ви також знайдете параметри дозволів. Вони визначають, які дії бот може виконувати на сервері. Наприклад, можна дозволити надсилання повідомлень, редагування ролей, управління каналами тощо. Ці дозволи налаштовуються залежно від функціоналу, який ви хочете реалізувати. Наприклад, для бота, який слідкує за повітряними тривогами, потрібен доступ до текстових і, можливо, голосових каналів.

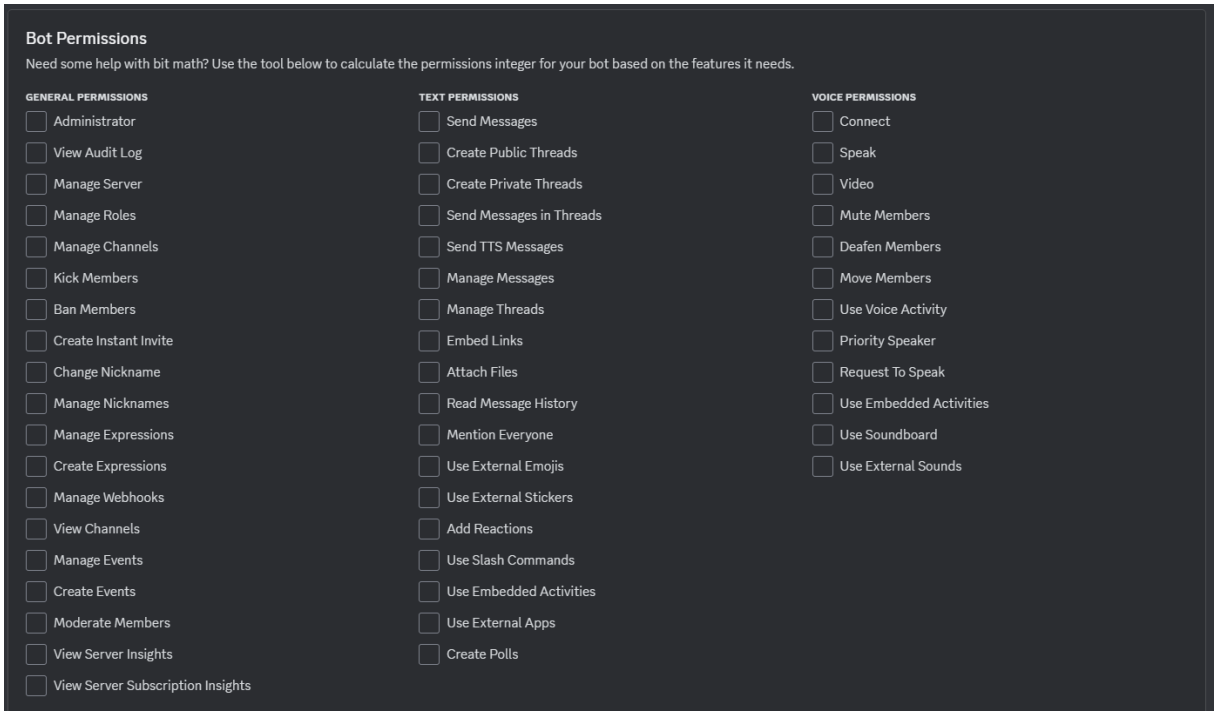


Рисунок 3.4. — Сторінка налаштування дозволів бота

Наступним важливим кроком є налаштування OAuth2 для додавання бота на сервер. Для цього у вкладці "OAuth2" необхідно відкрити "URL Generator". Тут потрібно вибрати об’єкт "bot", а також зазначити всі дозволи, необхідні для роботи бота. Після цього генерується спеціальний URL, який дозволяє запросити бота на потрібний сервер. Відкривши цей URL, ви отримаєте вікно з вибором сервера, куди хочете додати бота, після чого він з’явиться в списку учасників сервера (рис.3.5).

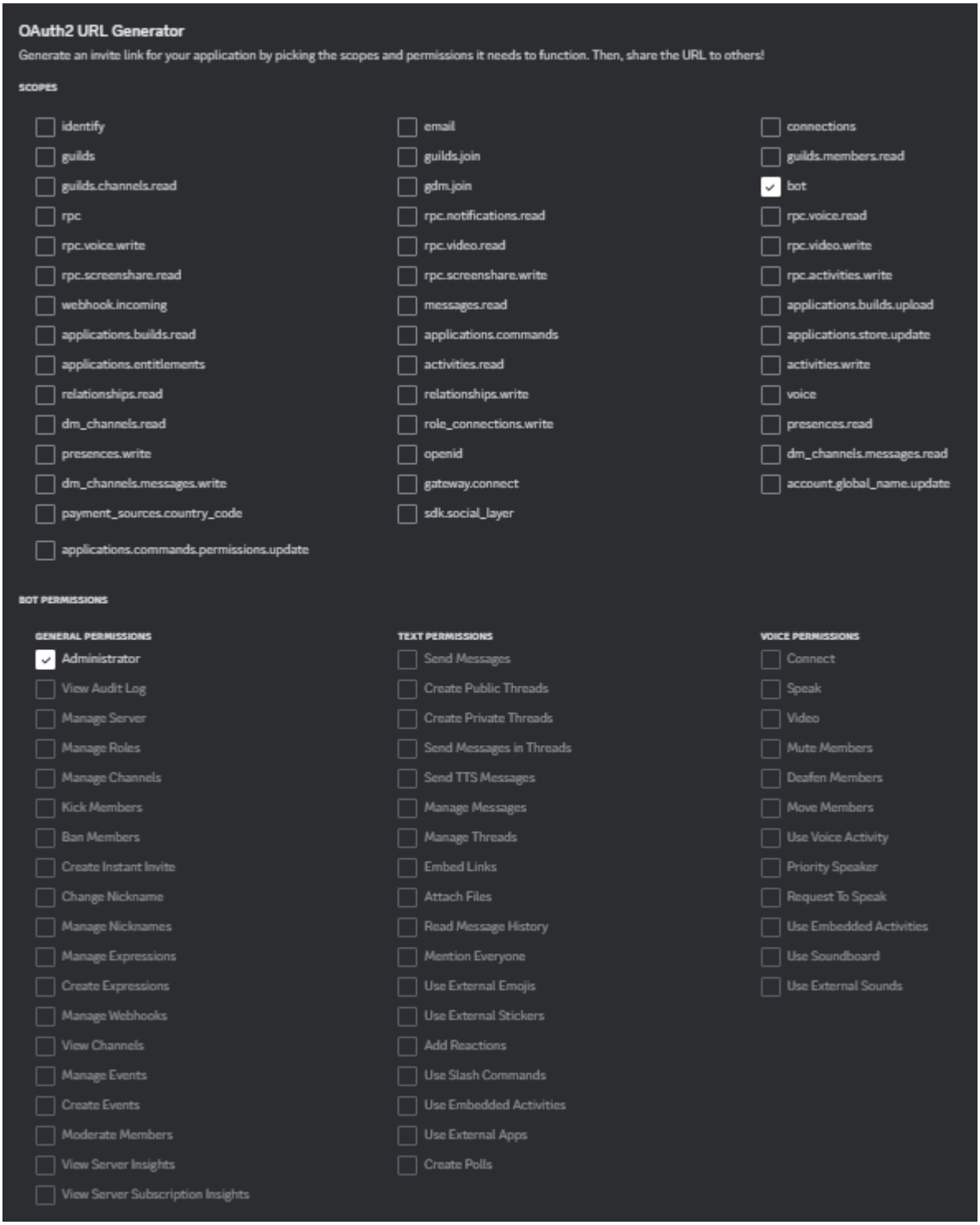


Рисунок 3.5. — сторінка генерації URL для додавання бота

Після того як бот доданий на сервер, можна переходити до його розробки. Для розробки на C# зручно використовувати бібліотеку Discord.NET, яка спрощує роботу з API Discord. Спочатку потрібно підключити бібліотеку до

проекту у вашому середовищі розробки, як було описано в розділі 2 , після чого налаштувати авторизацію бота через отриманий токен [24].

Під час програмування важливо налаштувати базові функції, такі як обробка команд і взаємодія з текстовими повідомленнями. Наприклад, бот може реагувати на певні команди користувачів, надсилати повідомлення в текстові канали або відтворювати звук у голосових каналах. У разі інтеграції зовнішніх API, таких як сервіс для моніторингу повітряних тривог, необхідно реалізувати логіку для отримання даних із цих API та їх обробки.

Для роботи з API було обрано вбудований клас C# .Net — System.Net.HttpClient. За допомогою його методу GetAsync(string requestUri) можна з легкістю зробити запит до API повітряних тривог та отримати дані у вигляді рядку JSON (рис.3.6) .

Рисунок 3.6. — Запит на API повітряних тривог

Робота над ботом також включає тестування і налагодження. Для цього можна створити тестовий сервер у (рис. 3.7), де ви будете перевіряти всі функції бота. Під час тестування важливо перевірити коректність взаємодії з API, стабільність роботи та відповідність дозволів. Залежно від потреб, функціонал бота можна розширювати, додаючи нові команди, інтеграції чи автоматизацію процесів.

Рисунок 3.7. — Тестовий сервер для перевірки бота

Таким чином, ми отримали повністю налаштованого Discord бота, який готовий до подальшого використання та інтеграції з іншими сервісами. Завдяки детальним налаштуванням у Discord Developer Portal, вдалося створити функціональний каркас для бота, що включає необхідні параметри доступу та взаємодії з каналами і серверами.

Зокрема, були визначені права бота для читання і надсилання повідомлень, управління каналами та інші важливі налаштування, що дозволяють ефективно реагувати на події. Отриманий бот тепер має всю необхідну базу для подальшої реалізації сповіщення про повітряні тривоги, що є основною метою його створення.

3.2. Створення та налаштування нового проекту в Visual Studio

Створення нового проекту в Visual Studio є важливим кроком на початку розробки програмного забезпечення, що забезпечує створення базової структури для майбутньої програми.

Після запуску Visual Studio користувач потрапляє на головний екран (рис.3.8), де є кілька варіантів дій.

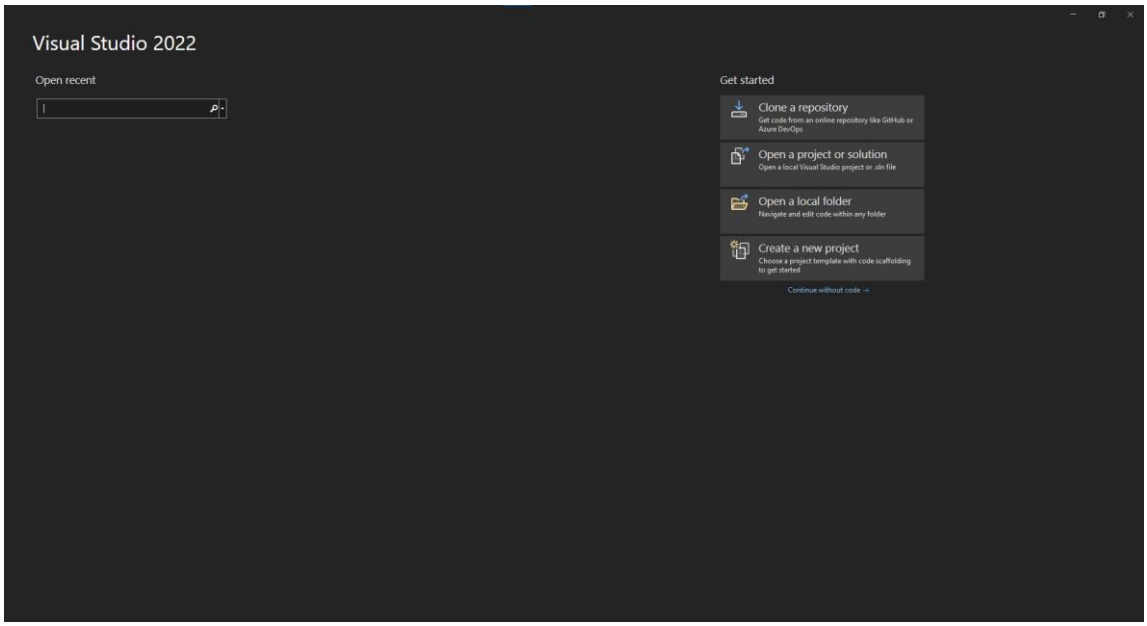


Рисунок 3.8. — Головне меню Visual Studio

Для створення нового проекту (рис.3.9) потрібно обрати опцію "Create a new project". Це відкриває інтерфейс, де представлено широкий вибір шаблонів проектів, таких як консольні додатки, веб-додатки, бібліотеки класів тощо[25].

Для розробки Discord бота зазвичай обирається шаблон "Console App (.NET Core)", оскільки він дозволяє працювати з необхідними інструментами для роботи з API.

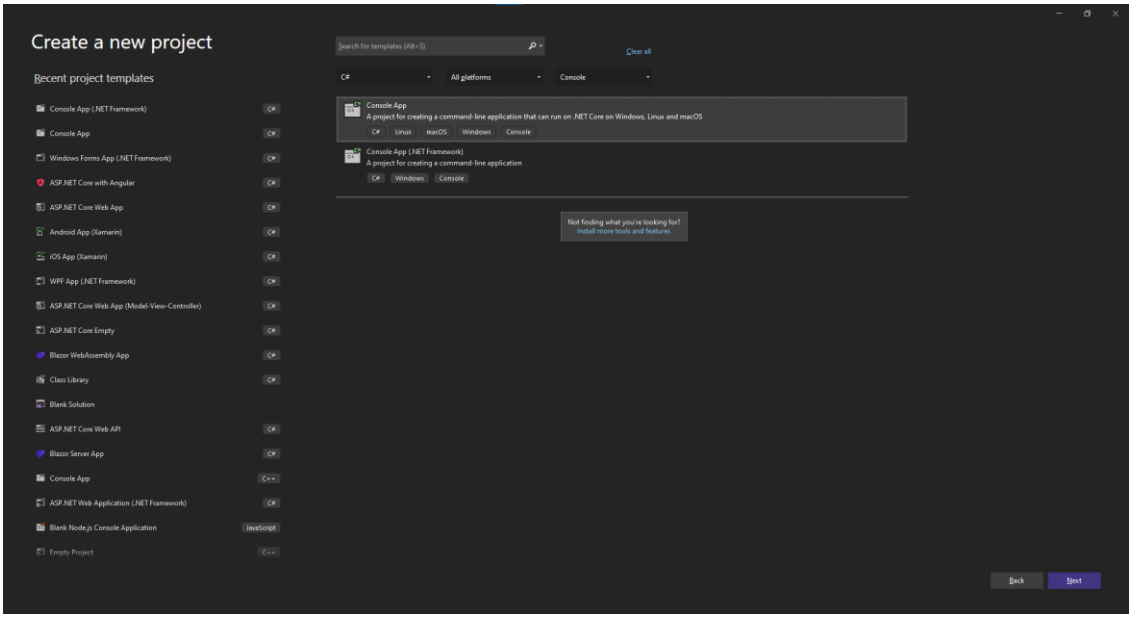


Рисунок 3.9. — Меню створення нового проекту Visual Studio

На наступному етапі відкривається вікно налаштувань проекту(рис.3.9). Тут потрібно вказати назву проекту, яка буде відображатися в середовищі розробки, а також місце його збереження на локальному диску. За бажанням можна задати ім’я рішення, якщо потрібно організувати кілька пов’язаних проектів. Після цього потрібно перейти до вибору цільової платформи .NET. Рекомендується обирати останню стабільну версію, наприклад, .NET 6.0 або .NET 7.0, оскільки вони забезпечують сучасний функціонал і високу продуктивність. Після підтвердження цих налаштувань система автоматично створює базовий проект.

Завершальний етап відкриває головний екран Visual Studio, де можна побачити створений файл Program.cs, що містить початковий код (рис.3.10). Цей файл забезпечує мінімальну структуру програми, яка може бути розширена для реалізації функціоналу Discord бота. На цьому етапі розробник може почати додавати необхідні залежності, бібліотеки та писати код для інтеграції з API[26].

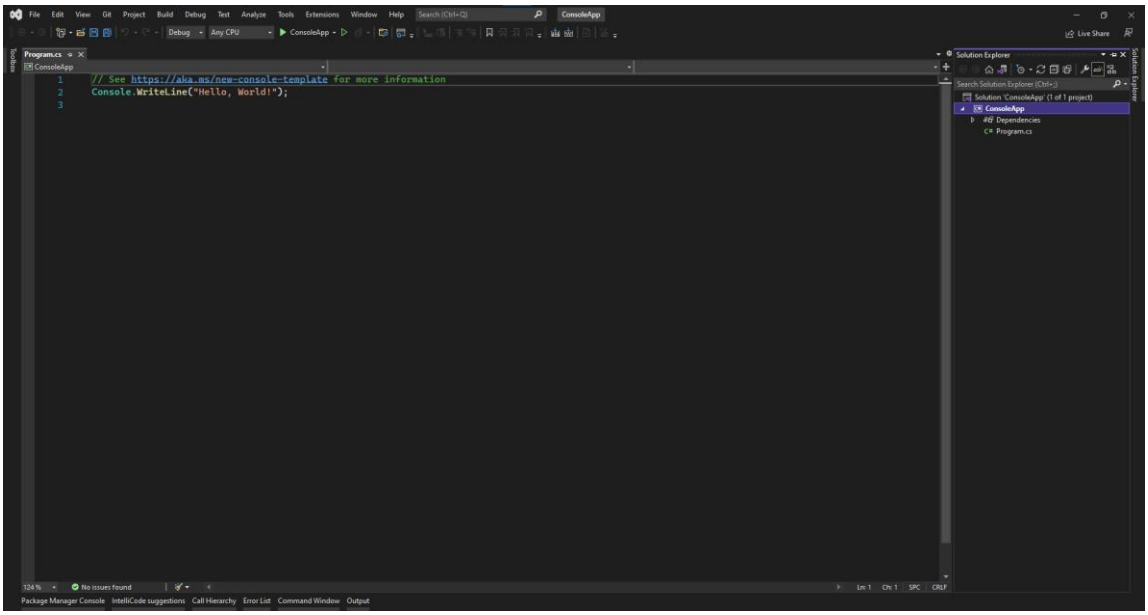


Рисунок 3.10. — Створений проект Visual Studio

Процес створення нового проекту в Visual Studio організований інтуїтивно і спрощує початок роботи над програмним забезпеченням. Сучасний інтерфейс, зрозумілі інструкції та широкий вибір шаблонів дозволяють ефективно розпочати розробку без додаткових налаштувань.

3.3. Розробка програмного забезпечення

Для створення базового Discord бота необхідно підготувати основну структуру програми. Це включає налаштування клієнта, який відповідатиме за взаємодію з платформою Discord, підключення до сервера за допомогою токена бота та налаштування обробки ключових подій, таких як отримання повідомлень, виконання команд або підключення до серверів.

У прикладі реалізації клієнт створюється за допомогою об'єкта DiscordSocketClient (рис.3.11) , який дозволяє боту взаємодіяти з Discord API. Для забезпечення роботи з усіма можливими типами подій використовується

конфігурація з параметром GatewayIntents.All. Це дозволяє боту відслідковувати повідомлення в чатах, реакції, оновлення статусів користувачів та інші події.

```
private static DiscordSocketClient _client;
0 references
public static async Task Main()
{
    var token = "[redacted]";

    _client = new DiscordSocketClient(new DiscordSocketConfig
    {
        GatewayIntents = GatewayIntents.All
    });
```

Рисунок 3.11. — Створення базового клієнту Discord Bot

Далі налаштовуються обробники подій (рис.3.12). Подія Log використовується для запису логів, що дозволяє розробникам відстежувати стан роботи бота. Обробник MessageReceived відповідає за перехоплення та обробку всіх повідомлень, отриманих у чатах. Подія Ready активується, коли бот успішно підключився до Discord сервера, і може бути використана для виконання початкових налаштувань або повідомлень. Крім цього, бот підтримує обробку слеш-команд через подію SlashCommandExecuted та багато інших.

```
_client.Log += Log;
_client.MessageReceived += MessageHandler;
_client.Ready += CommandBuild;
_client.SlashCommandExecuted += CommandHandler;
```

Рисунок 3.12. — Обробники подій в Discord

Щоб підключити бота до сервера, необхідно використовувати метод LoginAsync, який авторизує бота за допомогою токена, отриманого в *Discord Developer Portal*. Після цього бот запускається за допомогою методу StartAsync,

що дозволяє йому підтримувати активний зв’язок із сервером. Авторизація та запуск бота представлені на (рис.3.13).

```
await _client.LoginAsync(TokenType.Bot, token);
await _client.StartAsync();
```

Рисунок 3.13. — Авторизація та запуск бота

Щоб програма залишалась у стані виконання, використовується нескінченне завдання Task.Delay(-1), яке гарантує безперервну роботу (рис.3.14).

```
await Task.Delay(-1);
```

Рисунок 3.14. — Нескінчене завдання для роботи бота

Функція обробки команд CommandHandler

Функція CommandHandler відповідає за обробку слеш-команд, які надсилаються користувачами боту в Discord. Вона виконує перевірку активних повітряних тривог за допомогою API сервісу *alerts.in.ua* та повертає відповідь, якщо у визначеному регіоні, наприклад, в Автономній Республіці Крим, оголошено тривогу.

На початку функції оголошується змінна alertToken, яка зберігає токен для доступу до API сервісу *alerts.in.ua*. Це дозволяє програмі здійснювати запити до API для отримання актуальної інформації про повітряні тривоги (рис 3.15). Також визначається змінна channel, яка представляє канал у Discord, де була викликана команда.

Далі створюється об'єкт HttpClient, що використовується для виконання HTTP-запитів. Функція формує запит до API сервісу з використанням токена.

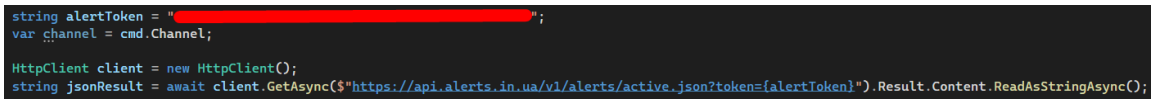


Рисунок 3.15. — Виконання HTTP-запита до API

Відповідь API, яка надходить у форматі JSON, зберігається у вигляді рядка після виконання методу GetAsync. Отримані дані десеріалізуються за допомогою бібліотеки Newtonsoft.Json у об'єкт типу AirRaidAlerts. Це перетворює JSON-структуру у зручний для роботи у програмі вигляд, який містить список актуальних тривог.

Подані класи є частиною реалізації структури даних для обробки інформації, отриманої від API сервісу *alerts.in.ua*, що надає дані про активні повітряні тривоги. Ці класи використовуються для десеріалізації JSON-відповіді від API у зручні для роботи об’єкти.

Клас AirRaidAlerts виконує роль контейнера (рис.3.16), що об’єднує всі дані, отримані від сервісу. Він містить масив об’єктів AirRaidAlert, які представляють конкретні записи про тривоги, а також об'єкт AirRaidMeta, який містить мета-інформацію про запит. Крім того, у класі є рядкове поле disclaimer, що може зберігати додаткову інформацію чи застереження, пов’язані з використанням API.

```
public class AirRaidAlerts
{
    public AirRaidAlert[] alerts;
    public AirRaidMeta meta;
    public string disclaimer;
}
```

Рисунок 3.16. — Клас AirRaidAlerts

Клас AirRaidMeta відповідає за зберігання мета-даних про оновлення інформації. Зокрема, він має два поля: last_updated_at, яке містить дату й час останнього оновлення даних, та type, що описує тип отриманих даних (рис.3.17).

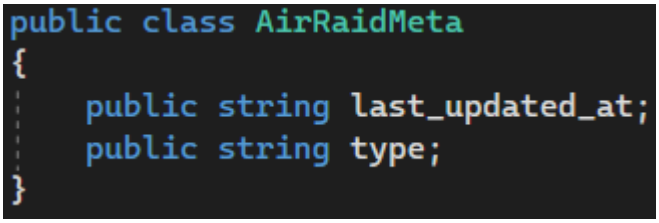


Рисунок 3.17. — Клас AirRaidMeta

Клас AirRaidAlert є ключовою структурою, що зберігає всі деталі про кожну окрему повітряну тривогу.

Поля цього класу включають (рис.3.18):

- id — унікальний ідентифікатор тривоги;
- location_title — назву локації, де була оголошена тривога;
- location_type — тип місця (наприклад, область чи район);
- started_at і finished_at — час початку та завершення тривоги;
- updated_at — час останнього оновлення цього запису;
- alert_type — тип тривоги (наприклад, повітряна, артилерійська тощо);
- location_uid — унікальний ідентифікатор локації;
- location_oblast та location_raion — область і район, де була оголошена тривога;
- notes — додаткові примітки до тривоги;
- calculated — поле, яке може містити додаткову інформацію, отриману шляхом обчислень.

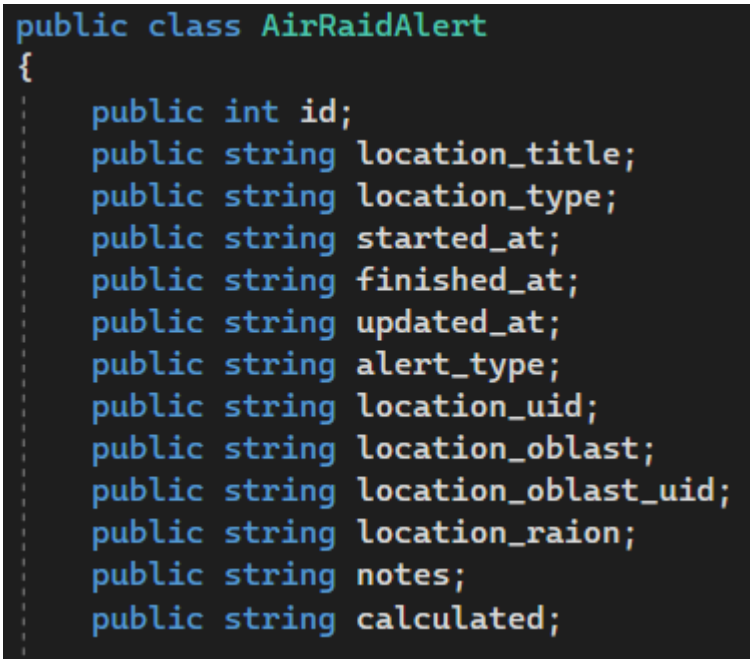


Рисунок 3.18. — Поля класу AirRaidAlert

Крім того, у класі AirRaidAlert визначено метод ToString(), який надає можливість представити дані про тривогу у зручному текстовому форматі. Це може бути корисним для налагодження або виведення даних для користувачів.

Разом ці класи забезпечують гнучкий і зручний спосіб роботи з даними про тривоги, що дозволяє легко інтегрувати API сервісу в програму для моніторингу та обробки тривог (рис.3.19).

```
public override string ToString()
{
    string str = string.Empty;

    str += "{\n";
    str += $"   \"id\": {id}\n";
    str += $"   \"location_title\": {location_title}\n";
    str += $"   \"location_type\": {location_type}\n";
    str += $"   \"started_at\": {started_at}\n";
    str += $"   \"finished_at\": {finished_at}\n";
    str += $"   \"updated_at\": {updated_at}\n";
    str += $"   \"alert_type\": {alert_type}\n";
    str += $"   \"location_uid\": {location_uid}\n";
    str += $"   \"location_oblast\": {location_oblast}\n";
    str += $"   \"location_oblast_uid\": {location_oblast_uid}\n";
    str += $"   \"location_raion\": {location_raion}\n";
    str += $"   \"notes\": {notes}\n";
    str += $"   \"calculated\": {calculated}\n";
    str += "}";

    return str;
}
```

Рисунок 3.19. — Функція ToString()

Після десеріалізації даних з JSON функція ітерує через список тривог за допомогою циклу foreach (рис.3.20). У кожному об’єкті типу AirRaidAlert перевіряється, чи відповідає його унікальний ідентифікатор регіону (в даному випадку location_uid) потрібному значенню, яке представляє Автономну Республіку Крим. Якщо тривога у цьому регіоні активна, бот відправляє повідомлення у канал, звідки була викликана команда, із текстом: "Повітряна тривога в Автономній Республіці Крим".

Ця функція демонструє взаємодію з API для отримання динамічних даних та їх аналізу з подальшою відповіддю користувачам. Завдяки такій реалізації бот може оперативно інформувати про повітряні тривоги у реальному часі, використовуючи команди у Discord.

```
alerts = JsonConvert.DeserializeObject<AirRaidAlerts>(jsonResult);

foreach (AirRaidAlert alert in alerts.alerts)
{
    if(alert.location_uid == "29")
    {
        await cmd.RespondAsync("Повітряна тривога в Автономній Республіці Крим");
    }
}
```

Рисунок 3.20. — Десеріалізація та ітерація по масиву повітряних тривог

Таким чином, ми отримали базову структуру бота, який підключається до Discord сервера, відслідковує події та готовий до подальшого розширення функціоналу. На цьому етапі бот може реагувати на основні події, такі як отримання повідомлень, і бути основою для реалізації складнішої логіки. Результат роботи функції CommandHandler наведено на рисунку 3.21.

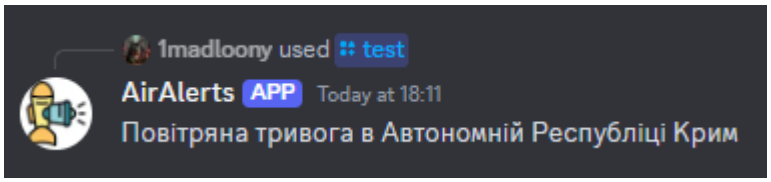


Рисунок 3.21. — Результат роботи функції CommandHandler

Висновки до третього розділу .

У цьому розділі було реалізовано ключові етапи створення програмного забезпечення для моніторингу повітряних тривог на платформі Discord. Ми досягли важливих результатів, які закладають основу для повноцінної роботи бота.

Перш за все, було створено та налаштовано обліковий запис бота у Discord через Discord Developer Portal. Це забезпечило доступ до унікального токену для авторизації та інтеграції бота з платформою Discord. Були визначені основні параметри бота та його доступ до необхідних серверів і каналів.

Далі було розгорнуто новий проект у середовищі розробки Visual Studio, яке було обрано через його багатий функціонал, зручність роботи з мовою програмування C# та підтримку необхідних бібліотек. Під час налаштування проєкту було створено структуру додатка, додано необхідні пакети та забезпечено інтеграцію з API Discord.

На завершальному етапі розділу було реалізовано основні функції бота. Створена програмна логіка включала обробку команд користувачів, взаємодію з API для отримання інформації про активні повітряні тривоги та її передавання у вигляді сповіщень до текстових каналів Discord. Впроваджено структуру даних для обробки відповіді від сервісу *alerts.in.ua*, що дозволило якісно обробляти та відображати актуальну інформацію.

Таким чином, у цьому розділі було створено та налаштовано програмний продукт, який готовий до виконання своєї основної функції — оперативного сповіщення користувачів про повітряні тривоги у зручному форматі через Discord.

ВИСНОВКИ

У процесі виконання дипломної роботи було створено програмний продукт — Discord бот для сповіщення про повітряні тривоги, який забезпечує швидке та ефективне інформування користувачів про виникнення небезпеки. Даний проект поєднує сучасні технології розробки програмного забезпечення та соціально значущу мету, що робить його важливим і актуальним у сьогоднішніх умовах.

На першому етапі роботи було проведено детальний аналіз предметної області. Вивчено основні функціональні можливості платформи Discord, яка забезпечує зручну взаємодію між користувачами через текстові та голосові канали. Також було досліджено, як можна використовувати Discord API для створення інтерактивних ботів, що взаємодіють із серверами платформи.

Було проведено аналіз різних підходів до роботи з API, зокрема REST і SOAP, дозволив зробити обґрунтований вибір на користь REST через його простоту та гнучкість у роботі з даними. Крім того, були оцінені існуючі приклади успішних ботів, таких як Soccer Guru, iTranslator та NotifyMe, що підтвердило доцільність та потенціал створення власного продукту для специфічної задачі.

На другому етапі було визначено ключові вимоги до програмного забезпечення та обрано інструменти, які забезпечили оптимальні умови для реалізації проекту. Мовою програмування було обрано C#, яка завдяки підтримці об’єктно-орієнтованого програмування (ООП) забезпечила зручність організації коду, модульність і можливість подальшого масштабування. Середовище розробки Visual Studio, завдяки своїм потужним інструментам для написання, тестування та дебагу коду, значно полегшило процес створення програмного забезпечення. Джерелом даних про повітряні тривоги було обрано сервіс *alerts.in.ua*, який надає актуальну інформацію через REST API у форматі

JSON. Для обробки цих даних застосовано бібліотеку Newtonsoft.Json, що дозволило ефективно працювати з відповідями сервісу.

Завершальний етап роботи був присвячений безпосередньо розробці програмного забезпечення. Створено базову структуру бота, який через Discord Developer Portal отримав унікальний токен для інтеграції з платформою Discord. Реалізовано механізм авторизації бота, а також основну програмну логіку, що включає обробку команд користувачів, отримання даних з сервісу *alerts.in.ua* та їх подальше передавання у текстові канали Discord. Для забезпечення точності та зручності роботи з даними впроваджено спеціалізовану структуру класів, яка дозволяє гнучко обробляти інформацію про повітряні тривоги, зокрема їх локацію, тип та час виникнення.

Результати виконаної роботи демонструють успішну реалізацію поставлених задач. Створений Discord бот не лише виконує свою основну функцію, а й має потенціал для подальшого розширення та вдосконалення. Його архітектура дозволяє інтегрувати додаткові функції, наприклад, підтримку багатомовності, розширені сповіщення чи інтеграцію з іншими сервісами.

Таким чином, дана робота не тільки вирішує поставлену технічну задачу, але й сприяє підвищенню рівня соціальної безпеки. Створений бот є інструментом, який може допомогти зберегти життя та здоров’я людей, оперативно інформуючи їх про небезпеку. Це підкреслює не лише технічну цінність проекту, але й його значущість для суспільства.

СПИСКИ СКОРОЧЕНЬ

API — application programming interface

ООП — об’єктно орієнтоване програмування

SOAP — Simple Object Access Protocol

REST — Representational State Transfer

HTTP — HyperText Transfer Protocol

HTTPS — HyperText Transfer Protocol Secure

URL — Uniform Resource Locator

JSON — JavaScript Object Notation

ID — identity document

XML — EXtensible Markup Language

JS — JavaScript

IoT — Internet of Things

RSS — Rich Site Summary

HTML — HyperText Markup Language

FTP — File Transfer Protocol

TCP — Transmission Control Protocol

UDP — User Datagram Protocol

SMPT — Simple Mail Transfer Protocol

ACID — atomicity, consistency, isolation, durability

WS — WebSocket

SSL — Secure Sockets Layer

DCOM — Distributed Component Object Model

CORBA — Common Object Request Broker Architecture

WSDL — Web Services Description Language

DM — Direct Message

GDM — Group Direct Message

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is Discord? [Електронний ресурс] — Режим доступу URL: https://discord.com/creators/what-is-discord
2. Martin Kleppmann, Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems, 1st Edition — 128-129 с.
3. Glenn Block, Pablo Cibraro, Pedro Felix, Howard Dierking, Darrel Miller, Designing Evolvable Web APIs with ASP.NET: Harnessing the Power of the Web 1st Edition, 12-13 с.
4. James Snell, Doug Tidwell, Pavel Kulchenko, Programming Web Services with SOAP: Building Distributed Applications, 1st Edition, 15 с.
5. James Snell, Doug Tidwell, Pavel Kulchenko, Programming Web Services with SOAP: Building Distributed Applications, 1st Edition, 21 с.
6. Leonard Richardson, RESTful Web APIs: Services for a Changing World: Episode 2: Short Sessions, 4 с.
7. Leonard Richardson, RESTful Web APIs: Services for a Changing World: Episode 2: Standardized Method, 8 с.
8. Leonard Richardson, RESTful Web APIs: Services for a Changing World: Episode 2: Resources and Representation: The Protocol Semantics of HTTP, 33-42с.
9. David Gourley, Brain Totty HTTP — The Definitive Guide David: Architceture Components of the Web, 17с.
10. Leonard Richardson, RESTful Web APIs: Services for a Changing World: 5 Domain- Specific Design: Maze + XML: A Domain- Specific Design, 59-74с.
11. Soccer Guru. [Електронний ресурс] — Режим доступу URL: https://soccerguru.live/

12. Discord Каталог програм iTranslator. [Електронний ресурс] — Режим доступу URL: https://discord.com/applicationdirectory/924619870891552799
13. NotifyMe. [Електронний ресурс] — Режим доступу URL: https://notifyme.bot/
14. Mark J Price, C# 12 and .NET 8 - Modern Cross-Platform Development Fundamentals - Eighth Edition: Start building websites and services with ASP.NET Core 8, Blazor, and EF Core 8. Chapter 5: Build Your Own Types with Object-Oriented Programming: Talking about OOP 231-291c.
15. Mark J Price, C# 12 and .NET 8 - Modern Cross-Platform Development Fundamentals - Eighth Edition: Start building websites and services with ASP.NET Core 8, Blazor, and EF Core 8. Chapter 6: Implementing Interfaces and Inheriting Classes: Talking about OOP 293-291 c.
16. Dan Clark, Beginning C# Object-Oriented Programming (Expert's Voice in .NET), Chapter 2: Designing OOP Solutions: Identifying the Class Structure 7 c.
17. Dan Clark, Beginning C# Object-Oriented Programming (Expert's Voice in .NET), Chapter 3: Modeling the Object Interaction 25c.
18. Dan Clark, Beginning C# Object-Oriented Programming (Expert's Voice in .NET), Chapter 5: Intoducing the .NET Framework and Visual Studio 59 c.
19. alerts.in.ua [Електронний ресурс] — Режим доступу URL: https://alerts.in.ua/
20. Lindsay Bassett, Introduction to JavaScript Object Notation: A To-the-Point Guide to JSON: What is JSON?: Key terms and Concepts, 4 c.
21. Lindsay Bassett, Introduction to JavaScript Object Notation: A To-the-Point Guide to JSON: 2 JSON Syntax: Syntax Validation, 10 c.

22. Lindsay Bassett, Introduction to JavaScript Object Notation: A To-the-Point Guide to JSON: 9 JSON on the Server Side: Serializing, Deserializing, and Requesting JSON, 87 c.
23. Newtonsoft. [Електроннйи ресурс] — Режим доступу URL: https://www.newtonsoft.com/json
24. Discord Developer Portal. [Електронний ресурс] — Режим доступу URL: https://discord.com/developers/docs/intro
25. Create a new project in Visual Studio [Електронний ресурс] — Режим доступу URL: https://learn.microsoft.com/en-us/visualstudio/ide/create-new-project?view=vs-2022
26. Install and manage packages in Visual Studio using the NuGet Package Manager [Електронний ресурс] — Режим доступу URL: https://learn.microsoft.com/en-us/nuget/consume-packages/install-use-packages-visual-studio